Departamento de computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Tesis de Licenciatura en Ciencias de la Computación

Metaheurística para el problema de Isomorfismo de Subgrafo con Pesos

Diego Barea (LU 320/97)
Emiliano Real (LU 281/97)

Director: Dr. Min Chih Lin

Resumen

El Isomorfismo de Subgrafo con Pesos es un problema de optimización combinatoria donde, dados un grafo A con pesos asociados a sus ejes y un grafo B utilizado como patrón, el objetivo es hallar el subgrafo de peso máximo entre todos los subgrafos de A isomorfos a B. En esta tesis proponemos una heurística para la búsqueda de "soluciones buenas" de manera eficiente basada en la metaheurística Tabu Search. Implementamos además la heurística propuesta y realizamos un análisis con el objetivo de estudiar la calidad de los de los resultados obtenidos.

Abstract

The Weighted Subgraph Isomorphism is a combinatorial optimization problem where, given a graph A which has weighted edges and a graph B used as pattern, the goal is to find the maximum weighted subgraph among all the subgraph of A isomorphic to B. In this thesis we introduce a heuristic for the search of "good solutions" in an efficient way based on the Tabu Search metaheuristic. We also implement the proposed heuristics and perform an analysis with the objective of studying the quality of the obtained results.

Índice

1 INTRODUCCIÓN 5

2 DEFINICIÓN Y COMPLEJIDAD DEL PROBLEMA 6

3 ESTADO DEL ARTE 8

4 BÚSQUEDA TABÚ 10

4.1 Introducción 10
4.2 Esquema general 11
4.3 Dimensiones de la Búsqueda Tabú 13

5 NUESTRO ALGORITMO 15

5.1 Entorno de la solución 15
5.2 Solución inicial 15
5.3 Periodo Tabú (Tabú Tenure) 17
5.4 Aspiración por defecto 18
5.5 Aspiración por influencia 19
5.6 Criterio de parada 20
5.7 Instancias del problema a utilizar 20
5.7.1 Transformación para peso de ejes positivo 20
5.7.2 Transformación para completar grafos 21
5.7.3 Problemas asociados a la transformación de instancias 22

6 ALGORITMO EXACTO Y COTA SUPERIOR 25

6.1 Solución al problema en forma exacta 25
6.2 Cota superior del peso de la solución 27
6.2.1 Algoritmo de cálculo 27
6.2.2 Ejemplo 28
6.2.3 Casos con pobre desempeño en el algoritmo de cálculo de cota 30

7 IMPLEMENTACIÓN 31

7.1 Archivo de configuración 31
7.2 Modo de ejecución 34

8 RESULTADOS 36

8.1 Introducción 36
8.2 Pruebas 36
8.3 Análisis de los Resultados 43
8.3.1 Efectividad de la Búsqueda Tabú 43
8.3.2 Porcentaje de mejora de Tabu Search sobre la heurística inicial 50
8.3.3 Porcentaje de mejora de Tabu Search sobre la heurística inicial según la dispersión de los pesos del Grafo Patrón 55
8.3.4 Análisis de tiempos 57

8.4 Casos especiales 59
8.4.1 Configuraciones utilizadas 59
8.4.2 Caso 1 – "Multi K_{10}" 59
8.4.3 Caso 2 – "Multi Ciclo" 61
8.4.4 Caso 3 – "Rueda de bicicleta" 63
8.4.5 Resultados 64

9 CONCLUSIONES 65

10 BIBLIOGRAFÍA 67

11 APÉNDICE A 69

11.1 Generador de grafos 69
11.2 Algoritmo de generación de grafos 70

12 APÉNDICE B 71

12.1 Efectividad de la heurística en Grafos de 9 a 12 nodos 71
12.2 Calidad de la cota en grafos de 9 a 12 nodos 72
12.3 Efectividad de la heurística en Grafos de 75 a 500 nodos 72

13 APÉNDICE C 73

13.1 Generación de instancia para la resolución del TSP 73
13.2 Configuraciones utilizadas - TSP 73
13.3 Casos de prueba – TSP 74
13.3.1 Ciclo de 11 Nodos 74
13.3.2 Ciclo de 100 Nodos 74

14 APÉNDICE D 76

15 APÉNDICE E 77

15.1 Transformación para peso de ejes positivo (Demostración) 77
15.2 Transformación para completar grafos (Demostración) 78
15.3 Fórmula para el cálculo de la distorsión 82

16 APÉNDICE F 85

16.1 Ejemplo de procesamiento del algoritmo exacto 85

1 Introducción

En este trabajo se presenta una heurística para tratar el problema de Isomorfismo de Subgrafo con Pesos. Se trata de un problema de optimización combinatoria donde dados un grafo A con pesos en sus ejes que llamaremos "Grafo Pesado" y un grafo B, con cantidad de nodos menor o igual a la de A, que llamaremos "Grafo Patrón", el objetivo es hallar un subgrafo de A isomorfo a B cuyo peso sea máximo.

Es un problema que pertenece a la clase NP-Hard, esto implica que en la práctica aun no se pudo hallar un algoritmo que lo resuelva (halle el valor óptimo) en tiempo polinomial. Un objetivo mucho más razonable es la búsqueda de un algoritmo aproximado de orden polinomial que para todas las instancias del problema halle una respuesta "cercana" al óptimo [4]. Esto motiva el desarrollo de un algoritmo que busque soluciones aproximadas en un tiempo mucho menor al que insumiría hallar el valor óptimo con un algoritmo exacto. Estas soluciones no necesariamente serán óptimas pero se buscará que sean razonablemente buenas.

Bajo este criterio el trabajo se centra en el desarrollo de una heurística basada en la metaheurística Tabu Search y en el análisis de la calidad de las soluciones obtenidas a través de la implementación de dicha heurística. En torno a estos dos objetivos, y no habiendo encontrado resultados previos contra los cuales comparar, realizamos como desarrollos complementarios la implementación de un algoritmo exacto y un algoritmo que calcula una cota superior. El primero orientado a hacer comparaciones entre las soluciones arrojadas por nuestra heurística y la solución óptima sobre instancias de tamaños reducido; el segundo como parámetro de calidad sobre instancias de gran tamaño donde, en general, no existen la posibilidad de conocer el valor óptimo en un intervalo de tiempo razonable.

Lo que queda del trabajo está dividido de la siguiente manera: en la sección 3 presentamos el estado del arte; en la sección 4 hacemos una introducción al Tabu Search[2, 8, 9]; en la sección 5 presentamos el algoritmo que desarrollamos y la manera en que se aplicó el método de Tabu Search [2, 8, 9]; en la sección 6 mostramos los desarrollos complementarios que hicimos del algoritmo exacto y del algoritmo de cálculo de cota superior; en la sección 7 hacemos un resumen de la implementación y de las diferentes opciones de configuración; en la sección 8 mostramos las pruebas realizadas y los resultados obtenidos y en la sección 9 damos nuestras conclusiones acerca de los resultados que obtuvimos, la efectividad de las diferente variantes y el trabajo en general, comentando también trabajos que serían interesantes de realizar en un futuro.

2 Definición y complejidad del problema

Tal como dijimos anteriormente, el problema de Isomorfismo de Subgrafo con Pesos es un problema de optimización combinatoria donde dados un grafo A con pesos en sus ejes que llamaremos "Grafo Pesado" y un grafo B, con cantidad de nodos menor o igual a la de A, que llamaremos "Grafo Patrón", el objetivo es hallar un subgrafo de A isomorfo a B cuyo peso sea máximo. La definición formal del problema es la siguiente:

Instancia: dos grafos, $A=(V_A, E_A, p:E_A \rightarrow \Re)$, $B=(V_B, E_B)$ / $|V_B| \leq |V_A|$, $S=\{S_A=(V, E)$ / S_A es subgrafo de A isomorfo a B$\}$ y una función $peso:S \rightarrow \Re$ definida como $\text{peso}(S_A(V,E)) = \sum_{e \in E} p(e)$.

Objetivo: Hallar $S_A \in S$ tal que $\text{peso}(S_A) \geq \text{peso}(S'_A)\ \forall\ S'_A \in S$.

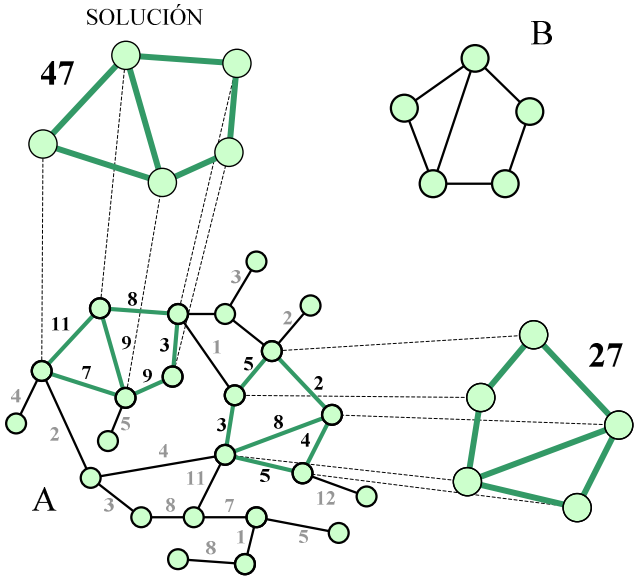


Figura 1 - Instancia del problema de Isomorfismo de Subgrafo con Pesos.

El problema de Isomorfismo de Subgrafo con Pesos pertenece a la clase NP-Hard y esto puede ser demostrado a través de su versión de decisión. Es fácil demostrar que la contraparte de decisión es NP-Completo. Esto se hace restringiéndolo al problema de Isomorfismo de Subgrafo, que ya sabemos que pertenece a la clase NP-Completo [4].Para ello se restringen las instancias a Grafos Pesados con ejes de peso 1 y se pregunta si existe un subgrafo S_A de $A=(V_A, E_A)$ isomorfo a $B=(V_B, E_B)$ cuyo peso sea mayor o igual a $k=|V_B|$. Podemos verificar en tiempo polinomial si a) efectivamente S_A es subgrafo inducido de A b) S_A es isomorfo a B y c) la suma de los pesos de las aristas de S_A es mayor o igual que k [4, 11].

Por otra parte, resolver el problema de decisión es equivalente a hallar el subgrafo de peso máximo y comparar el peso del grafo obtenido con k, lo cual indica que resolver dicho problema de decisión no es más difícil que resolver el problema de optimización. Esto último, sumado al hecho de que el problema de decisión es NP-Completo, constituye una prueba de que el problema de optimización es NP-Hard [4].

3 Estado del arte

Lamentablemente no hemos hallado trabajos que traten este problema con anterioridad. De todos modos podemos destacar que existen varios problemas NP-Completos conocidos en la literatura que son casos particulares de nuestro problema. Entre ellos podemos nombrar: Problema del Viajante de comercio (TSP), existencia de un conjunto independiente de tamaño K para cualquier K y existencia de un subgrafo completo de tamaño K para cualquier K (Clique Máxima).
En todos los casos se destaca que nuestro problema es una generalización de los problemas mencionados.

Para determinar la ***existencia de un subgrafo completo de tamaño K*** la transformación es sencilla y consiste en asignarle peso 1 a todos los ejes del Grafo Pesado y proponer un Grafo Patrón completo del tamaño deseado ***K***.
Se podrá comprobar la existencia de un subgrafo completo de tamaño ***K*** si el peso de la solución obtenida es igual a la cantidad de ejes del Grafo Patrón ($K \times (K-1)/2$), ya que el peso de cada eje es 1.

Un **conjunto independiente** de un grafo *G(V, E),* es un conjunto $I \subseteq V$ tal que para ningún par de vértices $x; y \in I$ ocurre que *x* e *y* son adyacentes en *G*.
Luego para determinar la ***existencia de conjunto independiente de tamaño K***, la transformación consistirá en:

- asignarle peso 1 a los ejes existentes en el Grafo Pesado
- agregar tantos ejes con peso 2 como sea necesario para completarlo
- El Grafo Patrón sería un grafo completo de tamaño ***K***.

Luego, tendremos un ***conjunto independiente de tamaño K*** si y solo si el peso de la solución obtenida es $K \times (K-1)$ *(el doble de la cantidad de ejes en G)**.***
Esto se debe a que en ese caso encontramos un subgrafo completo de tamaño ***K*** utilizando ejes que no pertenecían originalmente al grafo.

Para el ***problema del viajante de comercio*** (***TSP***), nuestro Grafo Patrón sería un ciclo de igual cantidad de nodos como ciudades se quieran visitar. En el Grafo Pesado original los nodos representarían las ciudades, y los ejes las distancias entre las mismas. Como nuestra solución busca un máximo y ***TSP*** un mínimo, entonces debemos invertir el peso de los ejes de alguna manera. Ejemplos de transformaciones para el Grafo Pesado son:

- Sea ***p*** el mayor peso de un eje, entonces a cada eje le reemplazamos su peso por ***p – peso_original***
- Sea ***p*** la sumatoria del peso de todos los ejes, entonces a cada eje le reemplazamos su peso por ***p – peso_original***

En cuanto a probables aplicaciones en la vida real vimos que el Isomorfismo de Subgrafo (sin pesos) es comúnmente utilizado en la Bioinformática. Un ejemplo de ello puede verse en [17]. El tema de nuestra tesis no se aplica en este problema en particular, sin embargo vemos en la Bioinformática un posible campo para la aplicación del Isomorfismo de Subgrafo con Pesos.

4 Búsqueda Tabú

4.1 Introducción

La Búsqueda Tabú (Tabu Search - **TS**) es una metaheurística conocida desde fines de los ochenta. Fue propuesta por Fred Glover en dos artículos aparecidos en la revista ORSA en 1989 [5,6].

El término tabú (taboo) proviene de la Polinesia, donde es usado por los aborígenes de la isla Tonga para referirse a cosas que no pueden ser tocadas porque son sagradas. Una acepción más moderna la define como "una prohibición impuesta por costumbres sociales como medida de protección", o algo "señalado como factor de riesgo". Esta definición es la que está más cerca de la esencia del método donde el riesgo a evitar es el de seguir un camino no productivo, incluyendo el de ser conducido a una trampa de la que no se puede salir (óptimo local) [8].

La Búsqueda Tabú es un procedimiento metaheurístico cuya característica distintiva es el uso de memoria adaptativa y de estrategias especiales de resolución de problemas.
El origen del **TS** aparece como resultado de la búsqueda de métodos que intentan cruzar cotas de factibilidad u optimalidad local tratadas como barreras en procedimientos clásicos, e imponer y eliminar cotas sistemáticamente para permitir la exploración de regiones no consideradas en otro caso.
La memoria adaptativa de **TS** utiliza la historia del desarrollo de la resolución del problema tomando decisiones sobre la búsqueda en base a cuatro factores principales: la propiedad de ser reciente, la frecuencia, la calidad, y la influencia [9].

TS es el origen del enfoque basado en memoria y estrategia intensiva en la literatura de las metaheurísticas, en contraposición con los métodos que no tienen memoria o que sólo usan una débil memoria basada en herencia. **TS** es también responsable de enfatizar el uso de los diseños estructurados para explotar los patrones históricos de la búsqueda, de forma opuesta a los procesos que confían casi exclusivamente en la aleatorización [9].

El destacable éxito de la Búsqueda Tabú para resolver problemas de optimización combinatoria difíciles, especialmente aquellos que surgen en aplicaciones del mundo real, ha causado una explosión de nuevas aplicaciones **TS** (ver Apéndice D) durante los últimos años [8].

4.2 Esquema general

La característica que distingue a la Búsqueda Tabú de las otras metaheurísticas de búsqueda es el uso de memoria la cual tiene una estructura basada en una lista tabú y en mecanismos de selección del siguiente movimiento.

Este método de búsqueda opera bajo el supuesto de que se puede construir un entorno donde se pueden identificar "soluciones vecinas" que pueden ser alcanzadas desde la solución actual.
El primer paso entonces es la generación de una solución inicial mediante una heurística constructiva. A partir de esta solución se realizan los movimientos buscando mejoras.

En la lista tabú se registran soluciones o movimientos que no deben ser tenidos en cuenta. Una forma sencilla de construir una lista tabú consiste en que cada vez que se realiza un movimiento, se introduce su inverso en una lista circular, de forma que los elementos de dicha lista están penalizados durante un período de tiempo.
Por lo tanto, si un movimiento está en la lista tabú no será aceptado, aunque aparentemente sea mejor solución que la solución actual.

Si bien mencionamos que la clasificación tabú se aplica sobre un movimiento, en realidad esta clasificación también puede ser aplicada sobre componentes más pequeñas de la solución denominadas ***atributos.*** Luego cada atributo puede ser clasificado como ***tabú activo*** o ***tabú inactivo***. Se denomina a esta clasificación como el ***estado tabú*** de un atributo.
Es importante tener presente que el hecho de que un atributo se halle en estado tabú activo no se corresponde con que un movimiento sea tabú. Más específicamente un movimiento puede contener atributos tabú activos pero no será tabú hasta que cierto conjunto de sus atributos se hallen en estado tabú activo.

El tamaño de la lista tabú ("tabu tenure" o "período tabú") es el tiempo o número de iteraciones que un elemento permanece en la lista tabú. Por ejemplo, si un movimiento ingresa a la lista con período tabú 3, permanecerá dentro de ella por 3 iteraciones.

A pesar de esto, la memoria cambiante del tabú puede tener "olvidos estratégicos", es decir que un elemento puede salir de la lista tabú antes de que se cumpla su plazo. Esto se implementa a través del Criterio de Aspiración. A modo de ejemplo, cuando un movimiento tabú resulta en una solución mejor que cualquiera visitada hasta el momento, su clasificación tabú puede ser reemplazada. Esta condición se llama ***criterio de aspiración por objetivo***.
El uso apropiado de los criterios de aspiración puede ser muy importante para permitir que un método **TS** proporcione sus mejores niveles de ejecución.

Una base para uno de los criterios de aspiración surge de introducir el concepto de influencia (***criterio de aspiración por influencia***), que mide el grado de cambio introducido en la estructura de solución. La influencia puede ser interpretada sobre distintas partes de la estructura de la solución y siempre se centra en la búsqueda de movimientos que introduzcan "grandes" cambios sobre el conjunto seleccionado.
A menudo el concepto de influencia es asociado a la idea de distancia de movimiento. Un movimiento será más influyente cuando alcance soluciones más lejanas a la solución actual.

Otro tipo de aspiración tiene como objetivo permitir que el algoritmo continúe. Es el caso la **aspiración por defecto**, que actúa cuando todos los movimientos son tabú y no hay ningún otro tipo de aspiración posible.

Las aspiraciones pueden ser de dos tipos: aspiración de movimiento y aspiración de atributo. Una aspiración de movimiento, cuando se satisface, revoca el estado tabú del movimiento mientras que una aspiración de atributo, cuando se satisface, revoca el estado tabú activo del atributo. En este último caso el movimiento puede no cambiar su clasificación tabú dependiendo de si la restricción tabú puede ser activada por más de un atributo.

Por último es necesario establecer las condiciones bajo las cuales el algoritmo finaliza su procesamiento. La técnica más comúnmente utilizada es controlar cuantas iteraciones transcurrieron sin notarse cambios relevantes. Aunque nunca debe faltar un tope a la cantidad de iteraciones que el algoritmo puede realizar [2, 8, 9].

Generar solución inicial $\mathbf{x_0}$
k := 1.
$x = x_0$. (x es la solución actual)
MIENTRAS la condición de finalización no se encuentre
    Identificar $N(x)$. (Vecindario de x)
    Identificar $T(x,k)$. (Lista Tabú)
    Identificar A(s,k). (Conjunto de Aspirantes)
    Determinar $N^*(x,k) = \{N(x) - T(x,k)\} \cup A(x,k)$. (Vecindario reducido)
    Escoger la mejor $x \in N^*(x,k)$
    "Guardar" x si mejora la mejor solución conocida $x_k := x$.
    Actualizar la lista tabú
    k := k+1.
FIN MIENTRAS

Figura 2 – Esquema general de la Búsqueda Tabú

4.3 Dimensiones de la Búsqueda Tabú

La memoria adaptativa de la Búsqueda Tabú utiliza la historia del desarrollo de la resolución del problema tomando decisiones sobre la búsqueda en base a cuatro dimensiones principales: la propiedad de ser reciente, la frecuencia, la calidad, y la influencia.

La dimensión de ***calidad*** hace referencia a la habilidad para distinguir el mérito de las soluciones. Es un concepto más amplio que el implícitamente usado en los métodos de optimización, en los cuales se considera que un movimiento es de mejor calidad que otro porque produce una "mejora". Bajo el enfoque de Búsqueda Tabú un movimiento puede ser de mejor calidad si, por ejemplo, su frecuencia de ocurrencia en el pasado es baja o no ha ocurrido antes y nos permite explorar nuevas regiones. La definición de calidad de una solución es flexible y puede ser adaptada a la naturaleza del problema.

La ***influencia*** mide el grado de cambio inducido en la estructura de solución.
El impacto que ciertos elementos o atributos tienen sobre una solución nos brinda información adicional que puede ser usada para privilegiar los movimientos etiquetados como influyentes.
El concepto de influencia suele ser asociado a la idea de distancia de movimiento. Un movimiento será más influyente cuando alcance soluciones más lejanas a la solución actual.
Los criterios de aspiración por influencia permiten que los movimientos influyentes puedan salir de la lista tabú antes del plazo establecido en su período tabú (tabú tenure).

La memoria basada en lo ***reciente*** (memoria de "corto plazo") constituye una forma de exploración agresiva que intenta realizar el mejor movimiento posible sujeto a restricciones para evitar que se reviertan o repitan ciertos movimientos. Es una manera de definir el entorno o vecindario reducido de una solución.
En la "memoria" se almacenan los últimos movimientos realizados y el objetivo primordial es hacer que la técnica de búsqueda pueda ir más allá de la optimalidad local. Los movimientos tabú no serán aceptados durante un cierto tiempo o un cierto número de iteraciones, tras lo cual se considera que la búsqueda está en una región distinta y pueden dejar de clasificarse como tabú dichos movimientos.
Sin estas restricciones, el método podría moverse hacia un punto fuera de un óptimo local, pero inmediatamente después caería en el mismo punto al determinarse que el mejor movimiento posible desde esa posición es precisamente el óptimo local antes mencionado.

La memoria basada en ***frecuencia*** (memoria de "largo plazo") registra la frecuencia de ocurrencia de los movimientos, las soluciones o sus atributos y aporta un nivel de aprendizaje más a ser utilizado en la elección de movimientos.

La memoria a largo plazo tiene dos estrategias asociadas, intensificación y diversificación.

La intensificación consiste en regresar a regiones ya exploradas para estudiarlas más a fondo. Para ello se favorece la aparición de aquellos atributos con alta frecuencia en las buenas soluciones encontradas.

Para evitar regresar a óptimos locales cada cierto número de iteraciones, la Búsqueda Tabú utiliza además otra estrategia, como es la diversificación, la cual consiste en visitar nuevas áreas no exploradas del espacio de soluciones. Para ello se modifican las reglas de elección para incorporar a las soluciones atributos que no han sido usados frecuentemente [2, 8, 9].

5 Nuestro algoritmo

En esta sección mostraremos en forma detallada el algoritmo de Búsqueda Tabú propuesto por nosotros.

5.1 Entorno de la solución

El método de Búsqueda Tabú opera bajo el supuesto de que se puede construir un entorno donde se pueden identificar "soluciones vecinas" que pueden ser alcanzadas desde la solución actual. Con el fin de facilitar la identificación de dichas "soluciones vecinas", nuestro algoritmo opera sobre instancias donde el Grafo Pesado es completo.
Más adelante mostraremos que a pesar de esta restricción, cualquier instancia del problema puede ser transformada para poder ser utilizada por nuestro algoritmo.

Hemos adoptado entonces como método de construcción del entorno un recurso que se utiliza frecuentemente: el intercambio de pares de nodos. El hecho de que el Grafo Pesado sea completo nos garantiza que cualquier intercambio de nodos nos mueve de una solución a otra.
Asociado a cada intercambio hay un valor de movimiento que representa el cambio en el valor de la función objetivo - en nuestro caso el peso de la solución - como resultado del intercambio propuesto.

Es interesante observar que bajo este esquema, para un Grafo Patrón de ***j*** nodos y un Grafo Pesado de ***k*** nodos el entorno de la solución en cada iteración estará constituido por $j \times (k - j) + j \times (j - 1)/2$ soluciones vecinas:

$j \times (k - j)$	Intercambios formados entre nodos que están en la solución contra nodos que no están en la misma
$j \times (j - 1)/2$	Intercambios formados entre nodos de la solución

5.2 Solución inicial

Como ya hemos mencionado, la *Búsqueda Tabú* trabaja partiendo de una solución inicial. Para generar la misma, desarrollamos un algoritmo goloso.
El hecho de que el Grafo Pesado sea completo nos asegura que cualquier solución generada mapeando cada nodo del Grafo Patrón con uno distinto del Grafo Pesado es una solución inicial válida.

Nuestro algoritmo goloso construye una solución a través de mapeos de nodos que siguen dos pautas básicas:

1) Si el nodo del Grafo Patrón a mapear NO tiene ejes adyacentes en común con otro nodo ya mapeado, entonces, siendo *k* el grado dicho nodo lo mapeamos

contra un nodo en el Grafo Pesado donde los *k* ejes adyacentes más pesados logren el máximo peso.

2) Si el nodo del Grafo Patrón a mapear SI tiene ejes adyacentes en común con otros nodos ya mapeados, entonces lo mapeamos contra un nodo en el Grafo Pesado donde la suma del peso de sus ejes sea máxima pero solo teniendo en cuenta aquellos ejes adyacentes en común con nodos ya mapeados.

A = Grafo Pesado

B = Grafo patrón

1) Se toma el nodo de mayor grado de ***B***.
   En caso de haber más de uno, se elige uno al azar.

2) Δ_B = grado del nodo obtenido en (1).

3) Se toma el nodo de ***A*** donde la suma de los Δ_B ejes más pesados relacionados a dicho nodo sea la mayor.
   En caso de haber más de uno, se elige uno al azar.

4) Mapeo el nodo obtenido en (1) con el obtenido en (3).

Mapeo del resto del grafo…

5) Para cada nodo de ***B*** (Recorriendo primero los de mayor grado)
   - Si el nodo en proceso no se relaciona a ninguno de los ya recorridos:
     - Mapeo con nodo de ***A*** (no mapeado), de la misma manera que se hizo en la inicialización. Esto es, sea ***k*** el grado del nodo de ***B*** en proceso, buscamos un nodo de ***A*** (no mapeado previamente) donde la suma de los ***k*** ejes más pesados relacionados a dicho nodo sea la mayor. En caso de haber más de uno, se elige uno al azar.
   - Si el nodo en proceso tiene ejes compartidos con nodos de ***B*** ya recorridos:
     - Se busca la lista de nodos de ***A*** que se mapearon con los nodos de ***B*** que comparten ejes con el nodo en proceso.
     - Luego se busca el nodo de A (no mapeado) donde la suma del peso de sus ejes relacionados con los nodos de la lista encontrada sea máxima.
     - Luego se mapea el nodo en proceso con el nodo de A encontrado.

Figura 3 - Algoritmo de búsqueda de la solución inicial

De esta manera se pretende incluir en la solución la mayor cantidad de ejes con pesos "elevados". Veamos un ejemplo de procesamiento:

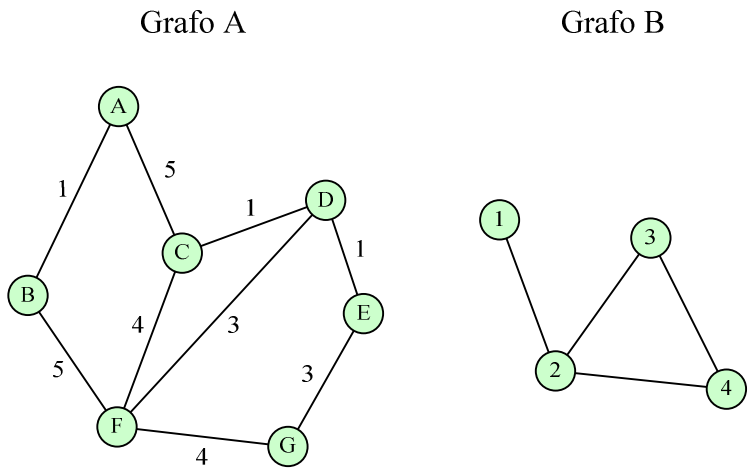


Figura 4 - En el gráfico se omiten los ejes de peso 0 del Grafo Pesado A.

El procesamiento comienza mapeando los nodos de mayor grado del Grafo Patrón, por ende en la primera iteración se buscará mapear el nodo **2**. Como es el primer mapeo seguimos la "pauta 1" y mapeamos contra el nodo **F** del Grafo Pesado.
En la segunda iteración, siguiendo la "pauta 2" mapeamos el nodo **3** con el **B**.
Luego, utilizando la misma pauta mapeamos el nodo **4** con **C** y finalizamos de igual manera mapeando **1** con **G**.

El peso de la solución inicial encontrada (2-F , 3-B , 4-C y 1-G) es 13 mientras que la solución óptima sería 1-A , 2-C , 3-B y 4-F con un peso de 14.

5.3 Periodo Tabú (Tabú Tenure)

Cada vez que la Búsqueda Tabú elige un intercambio, este pasa al estado ***tabú activo*** y se le es asignado un ***Periodo Tabú (PT),*** el cual nos indica la cantidad de iteraciones durante las cuales dicho intercambio no puede ser escogido nuevamente.

Consideramos un periodo tabú máximo a aquel que siempre deja un movimiento disponible o en estado tabú inactivo. El valor que se le asigna al **PT** es un porcentaje del máximo y es configurable.

Para un Grafo Patrón de ***j*** nodos y un Grafo Pesado de ***k*** nodos, el periodo tabú máximo es igual a $(j\times(k-j)+j\times(j-1)/2)-1$. Dicha ecuación la obtenemos a partir de la máxima cantidad de intercambios por iteración $(j\times(k-j)+j\times(j-1)/2)$ discutida en "Entorno de la solución".

El **PT** determina la cantidad de iteraciones durante las cuales un intercambio no puede ser escogido, con lo cual, sin mediar la aspiración por objetivo, y sin tener en cuenta ningún criterio de aspiración por influencia que modifique el estado tabú de un intercambio, pasarán $(j \times (k-j) + j \times (j-1)/2)$ iteraciones hasta que todos los intercambios pasen a ser tabú activo. Entonces el período tabú máximo proviene de restarle uno a esa cantidad.

Dependiendo de la frecuencia con la que un intercambio es elegido, el valor del **PT** se puede ver afectado. Esto quiere decir que penalizaremos a un movimiento con un **PT** mayor al configurado para tratar de evitar que se escojan siempre los mismos intercambios. Esta penalización por frecuencia también es configurable utilizando los siguientes parámetros:

- ***Frecuencia Tolerada (FT)***:
  La "Frecuencia de un intercambio media (FIM)", es la cantidad de veces que en promedio fue el elegido un intercambio (#iteraciones / #intercambios). Luego, en el archivo de configuración establecemos la "Frecuencia Tolerada", que indica que porcentaje sobre la "FIM" toleraremos sin penalizar.
- ***Penalización por frecuencia (PF):***
  La Penalización por frecuencia es el porcentaje con el cual se incrementa el periodo tabú de un intercambio en el caso de que la frecuencia del mismo supere lo establecido por la *Frecuencia Tolerada*.

El hecho de que el **PT** se configura como un porcentaje del máximo impedía que se puedan establecer valores que provoquen que en algún momento todos los intercambios sean tabú. Al agregar esta penalización por frecuencia esta característica se pierde, con lo cual implementamos criterios de aspiración por defecto para poder elegir un intercambio cuando todos los vecinos son tabú activo.

5.4 Aspiración por defecto

Cuando todos los intercambios son tabú y no hay ningún otro tipo de aspiración posible se asume la aspiración por defecto. Este tipo de aspiración tiene como objetivo permitir que el algoritmo continúe.

Para este tipo de aspiración hemos probado tres posibles implementaciones:

- Aspiración por defecto "menos tabú" **(APD-MT)**
  Si todos los movimientos son tabú, tomar aquel intercambio que le falten menos iteraciones para dejar de ser tabú.
- Aspiración por defecto "mejor intercambio" **(APD-MI)**

Si todos los movimientos son tabú, tomar el mejor intercambio, o el menos peor en caso de que todos los valores sean negativos

- Aspiración por defecto "sin penalización" **(APD-SP)**
  Si todos los movimientos son tabú, no se tiene en cuenta la penalización por frecuencia. De esta manera siempre habrá al menos un movimiento que no sea tabú.

La aspiración por defecto a utilizar es configurable.

5.5 Aspiración por influencia

Una base para uno de los criterios de aspiración surge de introducir el concepto de influencia, que mide el grado de cambio inducido en la estructura de solución.
En nuestra implementación utilizamos cuatro tipos distintos de aspiración por influencia, todos ellos centrados sobre los ejes de la solución:

Influencia Local
En este caso solo se consideran para determinar si un movimiento es o no influyente aquellos ejes adyacentes al nodo intercambiado que son adyacentes a nodos que pertenecen a la solución:

- Aspiración por Influencia por Peso Local: Se considera la influencia basándose en el peso que aporte el movimiento a la solución. Es decir, la suma de los pesos de los ejes adyacentes al nodo recientemente introducido siempre y cuando los mismos sean ejes adyacentes a nodos pertenecientes a la solución.
- Aspiración por Influencia por Ejes Local: Se considera la influencia basándose en la cantidad de ejes con peso mayor que 0 que aporte el movimiento a la solución. De nuevo, solo se consideran aquellos ejes que son adyacentes a nodos pertenecientes a la solución.

Influencia global
En este caso se consideran para determinar si un movimiento es o no influyente todos los ejes adyacentes al nodo recientemente intercambiado:

- Aspiración por Influencia por Peso Global: Se considera la influencia basándose en el porcentaje del peso del movimiento sobre el peso total del grafo, siendo el peso del movimiento la suma del peso de los ejes adyacentes al nodo que se quiere introducir en la solución. No importa que parte de este peso formará parte de la solución.
- Aspiración por Influencia por Ejes Global: Se considera la influencia basándose en el porcentaje de ejes del movimiento sobre el total de ejes con peso mayor a 0 del grafo, siendo los ejes del movimiento todos los ejes con peso mayor a 0 que son adyacentes al nodo que se quiere introducir en la solución. No importa cuales de estos ejes formarán parte de la solución.

La elección del tipo de aspiración por influencia a utilizar, así como también la no utilización de la misma, es configurable.

5.6 Criterio de parada

Gracias al criterio de aspiración por defecto, siempre habrá un intercambio que pueda ser utilizado por la Búsqueda Tabú. Por lo tanto, es necesario establecer las condiciones bajo las cuales el algoritmo finaliza su procesamiento.

Nuestra implementación del criterio de parada se basó en dos variables configurables. Una determina la cantidad máxima de iteraciones del proceso tabú mientras que la otra sirve para indicar cuantas iteraciones estamos dispuestos a esperar sin que el algoritmo de Búsqueda Tabú mejore la solución encontrada hasta el momento.

5.7 Instancias del problema a utilizar

Como ya planteamos en "Entorno de la solución", la Búsqueda Tabú opera bajo el supuesto de que se puede construir un entorno donde se pueden identificar "soluciones vecinas" que pueden ser alcanzadas desde la solución actual. Por tal motivo, establecimos que nuestro algoritmo operará sobre instancias donde el Grafo Pesado es completo.
Mencionamos también, que a pesar de esta restricción, cualquier instancia del problema puede ser transformada para poder ser utilizada por nuestro algoritmo. En un principio pensamos en una sola transformación polinomial que logre esto. Luego nos dimos cuenta que por claridad y simplicidad en las demostraciones era conveniente que la transformación que completa el grafo reciba instancias donde los ejes de dicho Grafo Pesado sean solo positivos. Por lo tanto terminamos utilizando dos transformaciones en forma anidada:

1) Transformaciones para peso de ejes positivo
2) Transformaciones para completar grafos

Primero encontraremos una transformación de la entrada de manera de pasar de una instancia general del problema a una instancia transformada, luego buscaremos una transformación de la salida, así poder interpretar el resultado que se obtuvo con dicha instancia transformada.
En el Apéndice E podremos luego apreciar las demostraciones que validan las transformaciones propuestas.

5.7.1 Transformación para peso de ejes positivo

Las siguientes transformaciones nos permiten pasar de una instancia cualquiera del problema, a otra instancia donde los ejes del Grafo Pesado A son solo positivos.

Transformación de la Entrada

Esta transformación consiste en sumar un mismo valor a cada uno de los ejes de manera que el peso de cada uno de ellos pase a ser positivo.

Sea ***I = {A(V,E,p), B(V_B, E_B)}*** la instancia de entrada, donde ***A*** es el Grafo Pesado con ***p*** su función de peso y ***B*** el Grafo Patrón.

Definimos la instancia transformada ***I' = {A'(V,E,p'), B(V_B, E_B)}*** donde
$p'(e) = p(e) + C$ y $C = 1 + \max(|p(e)|), \forall e \in E$

Nuestra nueva instancia ***I' = {A'(V,E,p'), B(V_B, E_B)}*** cumple entonces lo siguiente:

- $\forall e \in E, p'(e) > 0$

Transformación de la Salida

Sea ***G'($V_{G'}$,$E_{G'}$)*** la solución al problema utilizando la instancia de entrada transformada ***I' = {A'(V,E,p'), B(V_B, E_B)}***, entonces definimos ***G***, como la solución en base a la instancia original ***I*** de la siguiente manera:

- $G(V_G, E_G) = G'(V_{G'}, E_{G'})$

5.7.2 Transformación para completar grafos

Las siguientes transformaciones nos permiten pasar de una instancia donde los ejes del Grafo Pesado A son solo positivos, a otra instancia donde el Grafo Pesado A es completo y, adicionalmente, sus ejes son no negativos. De esta manera tendremos una instancia transformada que puede ser procesada por nuestro algoritmo.

Para satisfacer la condición que nos pide que la instancia de entrada tenga un Grafo Pesado A con ejes solo positivos basta con aplicar las transformaciones descriptas en el punto anterior.

Transformación de la Entrada

Esta transformación consiste en sumar un mismo valor a cada uno de los ejes ya existentes del Grafo Pesado para luego completar dicho grafo asignándole a cada nuevo eje peso 0.

Sea ***I = {A(V,E,p), B(V_B, E_B)}*** la instancia de entrada, donde ***A*** es el Grafo Pesado con ***p*** su función de peso, ***B*** el Grafo Patrón y $\forall e \in E, p(e) > 0$
Definimos la instancia transformada ***I' = {A'(V,E',p'), B(V_B, E_B)}*** donde

$$E' = E \cup \overline{E}$$

$$p'(e) = \begin{cases} 0 & \forall e \in \overline{E} \\ p(e) + C & \forall e \in E \end{cases}$$

con

$$C = \sum p(e) / e \in E$$

Nuestra nueva instancia ***I' = {A'(V,E',p'), B(V_B, E_B)}*** cumple entonces con la característica necesaria para que nuestro algoritmo la procese:

- ***A'*** es un grafo completo, pues $E' = E \cup \overline{E}$.

Adicionalmente obtenemos que $\forall e \in E', p'(e) \geq 0$.

Transformación de la Salida

Sea ***G'($V_{G'}$,$E_{G'}$)*** la solución al problema utilizando la instancia de entrada transformada ***I' = {A'(V,E',p'), B(V_B, E_B)}***, entonces definimos ***G*** como la solución en base a la instancia original ***I*** de la siguiente manera:

$$G(V_G, E_G) = \begin{cases} G'(V_{G'}, E_{G'}) & si \quad peso(G', p') \geq C \times \# E_B \\ \text{No existe} & si \quad peso(G', p') < C \times \# E_B \end{cases}$$

Donde

peso(***G',p'***)$= \sum p'(e) / e \in E_{G'}$

y, como ya especificamos, $C = \sum p(e) / e \in E$

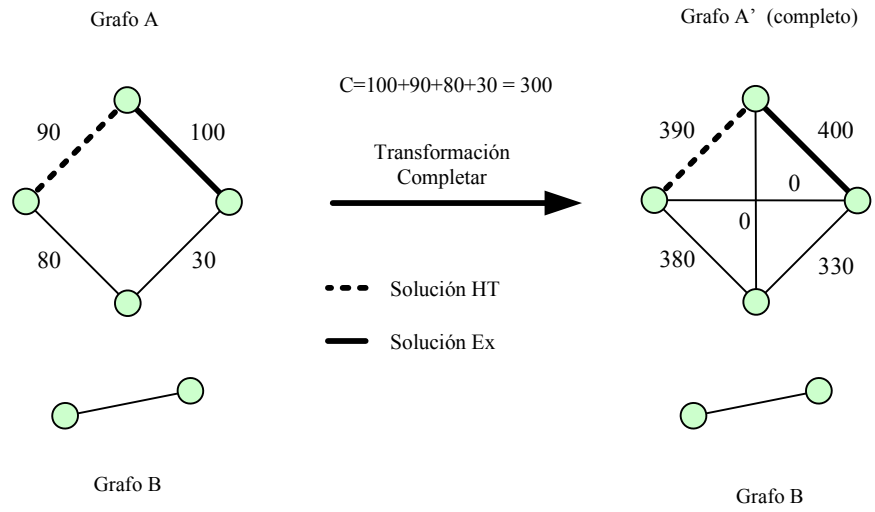
5.7.3 Problemas asociados a la transformación de instancias

Cuando se halla un resultado mediante una heurística podemos estimar en que medida el mismo se acercó al valor óptimo. Surge un problema cuando la instancia sobre la que se está trabajando fue obtenida mediante transformaciones. En estos casos, cuando no se alcanza el valor óptimo, un resultado de muy buena calidad sobre la instancia transformada puede no ser tan bueno sobre la instancia original.

El siguiente es un ejemplo simple donde, utilizando nuestra transformación para completar grafos, se observa claramente esta situación:

Sean,
A = G(V_A,E_A) el Grafo Pesado
B = G(V_B,E_B) el Grafo Patrón
HT = G(V_{HT},E_{HT}) la solución heurística

Ex = G(V_{Ex},E_{Ex}) la solución optima
Efe_HT_Ex = efectividad de la heurística sobre la instancia original
Efe_HT_Ex' = efectividad de la heurística sobre la instancia transformada
$C = \sum p(e) \ \forall e \in E_A$



p(HT) = 90

p(Ex) = 100

$$Efe\_HT\_Ex = \frac{p(HT)}{p(Ex)} = \frac{90}{100} = 90\%$$

p'(HT) = 390

p'(Ex) = 400

$$Efe\_HT\_Ex' = \frac{p'(HT)}{p'(Ex)} = \frac{390}{400} = 97{,}5\%$$

Figura 5 - Caso donde la efectividad sobre el grafo transformado es mayor que sobre el grafo original

La diferencia entre la efectividad de la heurística sobre la instancia transformada en comparación a la instancia original se debe a que dicha transformación genera una distorsión que se puede expresar formalmente de la siguiente manera:

Podemos expresar Efe_HT_Ex en función de Efe_HT_Ex' como:

$$Efe\_HT\_Ex = Efe\_HT\_Ex' \times \frac{p(HT)\left(\left|E_B\right| \times C / p(Ex) + 1\right)}{\left|E_B\right| \times C + p(HT)}$$

Donde:

p(HT): peso de la solución hallada por la heurística sobre la instancia original.
p(Ex): peso de la solución óptima sobre la instancia original.
C: constante de transformación (recordemos que $C = \sum p(e) \ \forall e \in E_A$)

Aplicado al ejemplo:

$$p(HT) = 90$$
$$p(Ex) = 100$$
$$\left|E_B\right| \times C = 1 \times 300 = 300$$
$$\Rightarrow Efe\_HT\_Ex = Efe\_HT\_Ex' \times \frac{90 \times (300/100 + 1)}{300 + 90} = \frac{360}{390} = 0{,}923$$

Luego

$$Efe\_HT\_Ex = 97{,}5\% \times 0{,}923 = 90\%$$

En el Apéndice E podemos apreciar la manera en que fue obtenida esta fórmula para el cálculo de la distorsión.

6 Algoritmo exacto y cota superior

En este capítulo describimos desarrollos complementarios que nos serán de utilidad para el análisis de la calidad de las soluciones obtenidas a través de nuestra heurística.
Implementamos un algoritmo exacto y un algoritmo que calcula una cota superior. El primero orientado a hacer comparaciones entre las soluciones arrojadas por nuestra heurística y la solución óptima sobre instancias de tamaños reducido; el segundo como parámetro de calidad sobre instancias de gran tamaño donde, en general, no existe la posibilidad de conocer el valor óptimo en un intervalo de tiempo razonable.
El hecho de no haber encontrado resultados previos contra los cuales comparar, hace que estas herramientas tengan una importancia significativa.

6.1 Solución al problema en forma exacta

Para la resolución del problema en forma exacta realizamos un algoritmo de fuerza bruta. Al momento de implementar, decidimos representar los grafos usando matrices de adyacencia.
Con esta representación utilizaremos una cantidad de espacios de memoria del orden de n^2 para guardar la información de los arcos (el programa utiliza matrices enteras ocupando exactamente n^2 posiciones pero podría utilizar media matriz para ocupar la mitad – de todos modos estamos dentro del orden cuadrático), y las operaciones relacionadas con el grafo implicarán, habitualmente, recorrer toda la matriz, con lo que el orden de las operaciones será $O(n^2)$. En cambio, con esta representación es muy fácil determinar, a partir de dos nodos, si están o no relacionados: sólo hay que acceder al elemento adecuado de la matriz y comprobar el valor que guarda.
Dicho valor, será el peso del eje para el caso del Grafo Pesado A. En cambio para el Grafo Patrón B, tendrá valores 0 (cero) o 1 (uno) para determinar si hay o no un eje entre dos nodos.
Usando matrices de adyacencia también encontramos una forma muy simple para saber el peso aportado por un mapeo dado entre los nodos del Grafo Patrón y los nodos de un subgrafo: se multiplican entre si todos los valores de ambas matrices que tengan igual fila y columna, luego de suman todos los resultados obtenidos. Ejemplo:

Subgrafo…

	0	1	2
0	-	3	6
1	-	-	10
2	-	-	-

Grafo Patrón…

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

Peso del mapeo: 3x1 + 6x0 + 10x1 = 13.
Otro detalle de implementación interesante es la forma en que se obtienen subgrafos. Para esto armamos un algoritmo que, teniendo en cuenta todos los índices de la matriz,

encuentra todos grupos “ordenados” de x cantidad de índices (donde x es la cantidad de nodos del subgrafo) e itera sobre estos.
Por ejemplo, si tenemos un Grafo Pesado de 4 nodos, entonces sus índices en la matriz serán 0, 1, 2 y 3. Luego, si nuestro Grafo Patrón es de 2 nodos, entonces necesitamos armar subgrafos de 2 nodos, con lo cual nuestro algoritmo nos devolverá los siguientes grupos de índices: (0,1), (0,2), (0,3), (1,2), (1,3) y (2,3).
Se debe notar que los diferentes grupos son “ordenados”, por ejemplo el grupo (1,0) no forma parte del resultado.
Una vez obtenidos estos grupos es muy fácil armar los subgrafos, ya que de la matriz inicial, nos quedamos solo con las filas y columnas indicadas por los índices y armamos nuevas matrices (los subgrafos).
Luego, para conseguir los diferentes mapeos que un subgrafo puede tener con el Grafo Patrón, permutamos entre si las filas y columnas de dicho subgrafo (si se cambia la fila 2 por la 3 también se cambia la columna 2 por la 3) y por cada permutación mapeamos uno a uno con los nodos de igual índice en el Grafo Patrón. Ejemplo:

Subgrafo…

	0	1	2
0	-	3	6
1	-	-	10
2	-	-	-

Permutaciones…

	2	0	1
2	-	6	10
0	-	-	3
1	-	-	-

	2	1	0
2	-	10	6
1	-	-	3
0	-	-	-

	0	2	1
0	-	6	3
2	-	-	10
1	-	-	-

	1	0	2
1	-	3	10
0	-	-	6
2	-	-	-

	1	2	0
1	-	10	3
2	-	-	6
0	-	-	-

```
A = Grafo Pesado
B = Grafo patrón
N_B = cantidad de nodos del Grafo B
mejor_sol_obtenida = vacío

Para todos los subgrafos de A que tengan N_B nodos
{
      Para cada mapeo entre los nodos del subgrafo y B
      {
            Si peso(solucion_obtenida)> peso(mejor_sol_obtenida)
            {
                  mejor_sol_obtenida = solucion_obtenida
            }
      }
}
Retornar mejor_sol_obtenida
```

Figura 6 - Algoritmo Exacto

En el Apéndice F podemos ver el funcionamiento del algoritmo en un caso particular.

6.2 Cota superior del peso de la solución

Si bien la salida del algoritmo exacto es de utilidad para analizar las soluciones obtenidas mediante la heurística, esta metodología solo se puede aplicar sobre grafos pequeños. Desarrollamos entonces un algoritmo de cálculo de la cota superior que nos permite estimar la calidad de las soluciones producidas por la heurística sobre grafos grandes.

6.2.1 Algoritmo de cálculo

Para obtener la cota recorremos el Grafo Patrón buscando mapearlo de la mejor manera posible con un nodo del Grafo Pesado pero tomando solo el grado de cada nodo del Grafo Patrón como única información. Es lógico pensar que la mejor manera de mapear un nodo de grado ***d*** sea justamente contra aquel nodo del Grafo Pesado donde la sumatoria de sus ***d*** ejes adyacentes más pesados sea la máxima.
Tomemos como ejemplo un nodo del Grafo Patrón cuyo grado es 3. Teniendo en cuenta solo este dato, sabemos que la mejor forma de mapearlo es contra el nodo del Grafo Pesado donde la sumatoria del peso de sus 3 ejes adyacentes más pesados sea máxima.
Una vez hallado dicho nodo, sumamos el peso de sus 3 ejes más pesados a la cota ya que sabemos que, en el mejor de los casos, ese nodo junto a esos 3 ejes formará parte de la solución.
La operación se repite con todos los nodos (b_1, b_2…,$b_{n'}$) del Grafo Patrón (B) teniendo en cuenta que un nodo (a_i) del Grafo Pesado (A) puede ser mapeado más de una vez. Esto es así ya que no sabemos cual de los nodos de B es mejor mapear contra el nodo a_i de A, debido a la simplificada información de la que disponemos.
En el único caso en que dado un nodo b_k no es necesario volver a mapearlo con un nodo a_i ya mapeado, a pesar de que sea éste la mejor opción, es cuando el grado b_k es igual al grado de algún b_x ya mapeado con a_i. En los casos que se da esta situación se intenta mapear con la segunda mejor opción.
Siguiendo el ejemplo anterior, si tomamos otro nodo del Grafo Patrón cuyo grado también es 3, lo intentaremos mapear contra el nodo del Grafo Pesado donde la sumatoria del peso de sus 3 ejes más pesados sea máxima. Pero dicho nodo, teniendo en cuenta sus 3 ejes más pesados, ya fue utilizado y no tiene sentido utilizarlo nuevamente.
Una vez recorrido todo el Grafo Patrón, obtenemos un valor formado por la suma del peso de varios ejes. Suponiendo que el mapeo fue perfecto, y suponiendo además que los ejes considerados en la cota son los que finalmente forman parte de la solución, entonces el peso de cada uno de dichos ejes fue contabilizado en dos ocasiones, una por cada nodo que conforman al mismo. O sea, el valor conformado hasta el momento es, en el mejor de los casos, exactamente el doble. Con lo cual dividimos por 2 dicho valor.

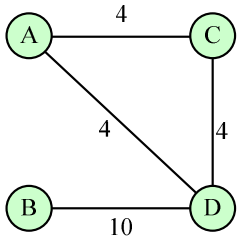
```
A = Grafo Pesado
B = Grafo patrón
Δ_B = máximo grado de los nodos de B
suma = 0

Para cada nodo v de A
      Para cada i = 1...Δ_B
            tabla(i,v) = suma de los i ejes más pesados del nodo v

Para cada nodo w de B
      valor = tabla(grado(w), v) tal que
                  tabla(grado(w), v) es máximo para todo nodo v de A
                  y tabla(grado(w), v) no fue usado
      marco tabla(grado(w), v) como usado
      suma = suma + valor

COTA = suma / 2
```

Figura 7 - Algoritmo de cálculo de la cota superior

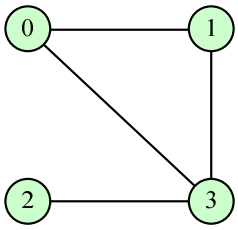
6.2.2 Ejemplo

A continuación mostraremos un ejemplo de procesamiento del algoritmo de cálculo de la cota superior.
Sean los siguientes grafos…

Grafo Pesado (A)



Grafo Patrón (B)



…se puede ver a simple vista que son dos grafos isomorfos, con lo cual el peso de la solución es simplemente la suma del peso de los ejes del grafo **A**, o sea 22.
La tabla usada por el algoritmo de cálculo de la cota sería la siguiente…

		Nodos de A			
		a	**b**	**c**	**d**
Grados de los nodos de B	**1**	**4**	**10**	**10**	**4**
	2	4+4=**8**	10+0=**10**	10+4=**14**	4+4=**8**
	3	4+4+0=**8**	10+0+0=**10**	10+4+4=**18**	4+4+0=**8**

Luego, iteramos sobre los nodos de **B** para mapearlos contra los nodos de **A,** armando así el peso nuestra cota…

Iteración 1: mapeo del nodo 2 de B		Nodos de A			
		a	**b**	**c**	**d**
Grados de los nodos de B	**1**	4	10	10	4
	2	8	10	14	8
	3	8	10	**18**	8

Iteración 2: mapeo del nodo 0 de B		Nodos de A			
		a	**b**	**c**	**d**
Grados de los nodos de B	**1**	4	10	10	4
	2	8	10	**14**	8
	3	8	10	18	8

Iteración 3: mapeo del nodo 1 de B		Nodos de A			
		a	**b**	**c**	**d**
Grados de los nodos de B	**1**	4	10	10	4
	2	8	**10**	14	8
	3	8	10	18	8

Iteración 4: mapeo del nodo 3 de B		Nodos de A			
		a	**b**	**c**	**d**
Grados de los nodos de B	**1**	4	**10**	10	4
	2	8	10	14	8
	3	8	10	18	8

Como se puede observar, tanto el nodo **b** como el nodo **c** de **A** son mapeados en más de una ocasión, ya que fueron la mejor opción de mapeo más allá de ya haber sido utilizados.
Se da una excepción en la tercera iteración, donde la mejor opción era el mapeo del nodo **1** (grado 2) con el nodo **c**, pero el mismo ya había sido utilizado en la segunda iteración, donde fue mapeado con otro nodo de **B** también de grado 2 (el nodo **0**).
Al existir un mapeo previo con otro nodo de grado 2 (o sea del mismo grado que el nodo actual a mapear) el algoritmo busca la segunda mejor opción que en este caso es el nodo **b**. Luego, el peso de la cota según nuestro algoritmo es:

$$\textbf{Cota} = (18 + 14 + 10 + 10)\ /2 = 26$$

Si en cambio no permitiésemos ningún tipo de mapeo repetido entonces nuestro algoritmo no estaría calculando la cota. Esto es fácil de ver utilizando este mismo ejemplo como podemos apreciar a continuación…

Iteración 1: mapeo del nodo 2 de B		Nodos de A			
		a	**b**	**c**	**d**
Grados de los nodos de B	**1**	4	10	10	4
	2	8	10	14	8
	3	8	10	**18**	8

Iteración 2: mapeo del nodo 0 de B		Nodos de A			
		a	**b**	**c**	**d**
Grados de los nodos de B	**1**	4	10	10	4
	2	8	**10**	14	8
	3	8	10	18	8

Iteración 3: mapeo del nodo 1 de B		Nodos de A			
		a	**b**	**c**	**d**
Grados de los nodos de B	**1**	4	10	10	4
	2	**8**	10	14	8
	3	8	10	18	8

Iteración 4: mapeo del nodo 3 de B		Nodos de A			
		a	**b**	**c**	**d**
Grados de los nodos de B	**1**	4	10	10	**4**
	2	8	10	14	8
	3	8	10	18	8

Podemos observar que en la 2° y 3° iteración no elegimos mapear los nodos de **B** con el nodo **c** de **A**, ya que dicho nodo ya había sido mapeado.
Lo mismo ocurre en la 4° iteración, donde se termina mapeando **d** con **3**, de manera de no repetir nodos mapeados. Luego, el peso de la cota según este algoritmo sería:

$$\textbf{Cota} = (18 + 10 + 8 + 4) / 2 = 20$$

Como el peso de la solución es 22, si el algoritmo no permitiese ningún tipo de mapeo repetido entonces no estaríamos calculando una cota superior.

6.2.3 Casos con pobre desempeño en el algoritmo de cálculo de cota

Hemos encontrado que la cota puede ser tan mala como uno quisiera. Con esto nos referimos a que podemos encontrar una familia infinita de Grafos Patrón y Grafos Pesado para los cuales a medida que la dimensión del grafo crece la razón entre el peso de la solución óptima y la cota tiende a infinito.
El caso que se muestra en la Figura 8 es un ejemplo donde la cota no es buena. En este caso el valor de la misma es 8 veces más grande que el peso de la solución.
Por otra parte, solo basta agregar un nivel a cada uno de los grafos (patrón y pesado) y un nuevo nodo con su correspondiente eje adyacente a cada uno de los nodos A y B para que la cota pase a ser aproximadamente 9 veces superior al valor óptimo.
Este procedimiento se puede repetir infinidad de veces obteniendo en cada iteración una razón ***cota/peso de la solución*** cada vez mayor.
Para mejorar esta situación modificamos el algoritmo de cálculo de cota de manera que devuelva el mínimo entre el resultado obtenido con el procedimiento descripto y la suma de todos los ejes del Grafo Pesado. De esta manera ponemos un límite al valor de la cota para aquellos casos donde el algoritmo tiene un pobre desempeño.

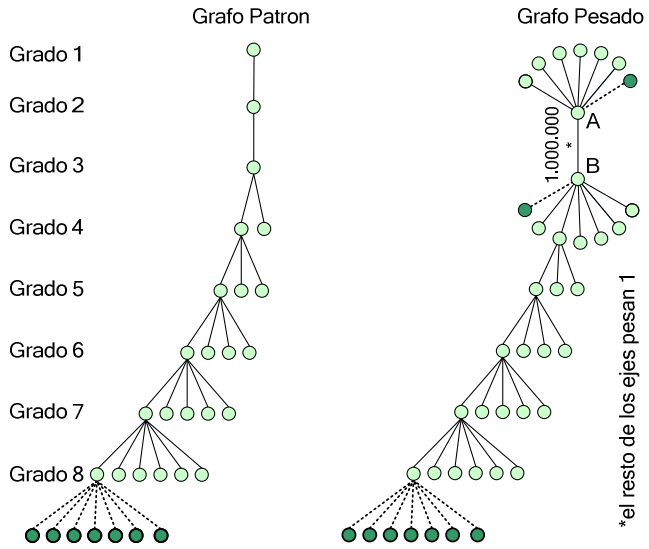


Figura 8 – Agregado de un nivel en un caso en el que el cálculo de la cota no es bueno

7 Implementación

En esta sección veremos las distintas opciones de ejecución que tiene el algoritmo que implementamos. El mismo toma algunos parámetros por la línea de comandos pero las principales variables se definen en un archivo de configuración.
Durante el desarrollo y las pruebas se utilizó un Pentium 4 de 3.4GHz con 1GB de memoria RAM. El sistema operativo sobre el cual se trabajó fue Windows XP Professional.
El programa que implementa la Búsqueda Tabú fue desarrollado utilizando el lenguaje C++ mientras que se utilizó Microsoft Visual Basic para el desarrollo de distintas herramientas complementarias ya sea para crear grafos (Apéndice A) como para analizar los resultados arrojados por las pruebas.
A continuación veremos un ejemplo del archivo de configuración que utiliza nuestro algoritmo. Luego mostraremos la manera en que se ejecuta el programa y los parámetros necesarios para dicha ejecución.

7.1 Archivo de configuración

Tal como detallamos en el capítulo "Nuestro algoritmo", tenemos varios parámetros configurables a través de un archivo de configuración, lo cual nos permite tener gran flexibilidad a la hora de realizar nuestras pruebas. Dicho archivo, el cual se llama "**tabu.ini**" y debe estar ubicado en la misma carpeta en la que se encuentra el ejecutable, incluye los siguientes parámetros:

- Periodo Tabú ("`PeriodoTabu`"):
  Consideramos un periodo tabú máximo como aquel que siempre deja un movimiento disponible o en estado tabú inactivo. El valor que se le asigna al periodo tabú en cada prueba se indica como un porcentaje del máximo.

- Aspiración por Influencia por Peso Global ("`API=1`"):
  Se considera la influencia basándose en el porcentaje del peso del movimiento sobre el peso total del grafo, siendo el peso del movimiento la suma de los pesos de los ejes adyacentes al nodo que se quiere introducir en la solución. No importa que parte de este peso pertenezca a la solución.

- Aspiración por Influencia por Ejes Global ("`API=2`"):
  Se considera la influencia basándose en el porcentaje de ejes del movimiento sobre el total de ejes mayor a 0 del grafo, siendo los ejes del movimiento todos los ejes con peso mayor a 0 que son adyacentes al nodo que se quiere introducir en la solución. No importa cuales de estos ejes formarán parte de la solución.

- Aspiración por Influencia por Peso Local ("`API=3`"):

Se considera la influencia basándose en el peso que aporte el movimiento a la solución.

- Aspiración por Influencia por Ejes Local ("`API=4`"):
  Se considera la influencia basándose en la cantidad de ejes con peso mayor que 0 que aporte el movimiento a la solución.

- Sin aspiración por influencia ("`API=0`").

- Frecuencia Tolerada ("`FrecuenciaTolerada`" **FT**) y Penalización por frecuencia ("`PenalizacionPorFrec`" **PF**):
  Se penaliza un movimiento incrementando su periodo tabú en **PF%** cuando su frecuencia de aparición es **FT%** mayor que la media.

- Aspiración por defecto: menos tabú ("`APD=1`"):
  Si todos los movimientos son tabú, tomar aquel intercambio que le falten menos iteraciones para dejar de ser tabú.

- Aspiración por defecto: mejor intercambio ("`APD=2`"):
  Si todos los movimientos son tabú, tomar el mejor intercambio, o el menos peor en caso de que todos los valores sean negativos.

- Aspiración por defecto: sin penalización ("`APD=3`"):
  Si todos los movimientos son tabú, no se tiene en cuenta la penalización por frecuencia. De esta manera siempre habrá al menos un movimiento que no sea tabú, ya que recordemos que **PT** se especifica como un porcentaje (menor a 100) del máximo periodo tabú, el cual por definición, es aquel que siempre deja un movimiento disponible o en estado tabú inactivo.

- Máxima cantidad de iteraciones **("`MaxIter`"):**
  Cantidad de iteraciones máxima que realiza el algoritmo tabú.

- Máxima cantidad de iteraciones sin mejora **("`MaxIterSinMejora`"):**
  Cantidad de iteraciones máxima que realiza el algoritmo tabú sin que mejore el peso de la solución.

A continuación mostramos un ejemplo del archivo de configuración:

```
# Cantidad de iteraciones máxima que realiza el algoritmo tabú
MaxIter=1000
# Cantidad de iteraciones máxima sin que mejore el peso de la solución
MaxIterSinMejora=3000
```

```
# El periodo tabú puede también ser pasado por línea de comandos.
# En caso de que sea así, tendrá prioridad por sobre el valor
# asignado en este archivo de configuración.
# En "PeriodoTabu" hay que poner que porcentaje del máximo
# periodo tabú posible queremos usar
PeriodoTabu=100   # ingresar solo valores enteros

# En "PenalizacionPorFrec" hay que indicar el % del periodo tabú
# que utilizaremos en caso de penalizar por frecuencia.
# Por ejemplo si el periodo tabú es 10 ( <-- no % sino real )
# y la penalización es 20%, cuando debido a la frecuencia
# del intercambio hay que penalizar, el periodo tabú que se le
# asigna al intercambio será de 12 ( 10 + 20% de 10 ) y no 10.
PenalizacionPorFrec=0   # ingresar solo valores enteros

# Frecuencia de un intercambio (FI) = cantidad de veces que fue
# elegido un intercambio.
# Frecuencia de un intercambio media (FIM) = cantidad de veces
# que en PROMEDIO fue el elegido un intercambio,
# (#iteraciones/#intercambios)
# En el campo "FrecuenciaTolerada" indicaremos que porcentaje
# sobre la "FIM" toleraremos sin penalizar.
# Por ejemplo, si la "FIM" es 10 y el intercambio que acabamos
# de usar tiene una "FI" de 12, entonces está un 20% por encima
# de la media.
# Ese intercambio será penalizado solo si la "FrecuenciaTolerada"
# es menor a 20.
FrecuenciaTolerada=100  # ingresar solo valores enteros

# En "Estadistica" indicar con 'S' si queremos estadística y
# con 'N' en caso de que no
Estadistica=S

# CRITERIOS DE ASPIRACION (hacer tabú inactivo un intercambio
# tabú para luego evaluar si lo elegimos)
# Aspiración por objetivo ( el intercambio no es tabú porque
# da la mejor solución hasta ahora ) - implementado
# Aspiración por influencia ( los intercambios influyentes,
# no se consideran tabú y son evaluados)
#     opción 1: influencia por peso global ( según el peso
#               de los nodos que se intercambian )
#     opción 2: influencia por cantidad de ejes global ( según
#               cantidad de ejes con peso mayor a 0 de los nodos que
#               se intercambian )
#     opción 3: influencia por peso ( s/ el peso de los nodos
#               que se intercambian solo usando peso de ejes
#               de la solución)
#     opción 4: influencia por cant. de ejes ( s/ cant. de ejes con
#               peso mayor a 0 que aportan a la solución los nodos que
#               se intercambian )
```

```
#     Se puede utiliza más de una opción, separando con ";".
#     Por ejemplo "API=1;2"
API=1
# Si no se la asigna ningún valor a "API", entonces no
# habrá aspiración por influencia

# Aspiración por defecto ( todos los intercambios son tabú y
# no hay ningún otro tipo de aspiración posible)
#     opción 1: Si todo es tabú, tomar el intercambio menos tabú
#     opción 2: Si todo es tabú, tomar el mejor intercambio,
#               o el menos peor en caso de que todos los valores
#               sean negativos
#     opción 3: Si todo es tabú, no tener en cuenta la penalizaciones,
#               de esta manera si o si habrá un movimiento NO tabú
APD=1  # valor ADP default es 3
```

7.2 Modo de ejecución

Para ejecutar el programa se deberá especificar el Grafo Pesado, el Grafo Patrón, el archivo donde se dejará la solución y opcionalmente un flag (-e) para determinar si queremos que la solución muestre un detalle de las iteraciones donde hubo mejoras.
La ejecución en línea de comandos tendrá entonces esta forma:
TabuEXE GrafoPesado.adg GrafoPatron.adg Solucion.sol –e

Los grafos se guardan en archivos de la siguiente manera:

```
#Fecha 18/08/2005 20:42:28
#Las Dimensiones: #Nodos #Ejes
6 6
#Los Nodos: Nodo X Y (Ubicación en la pantalla)
0 8317 3673
1 6817 1074.924
2 3817 1074.924
3 2317 3673
4 3817 6271.076
5 6817 6271.076
#Los Ejes: Nodo1 Nodo2 Peso
0 4 94210.8750343323
1 2 81389.4748687744
1 3 85236.656665802
1 5 17412.7340316772
2 3 61348.4263420105
2 5 25125.9803771973
```

Si estuviéramos intentando leer el grafo para utilizarlo como Grafo Pesado se asumirán ejes de peso cero en caso de que dichos ejes no estén especificados en el archivo. Esto es así debido a que como ya mencionamos, nuestro algoritmo trabaja con Grafos Pesados completos.

Si en cambio, estuviéramos intentando leer un Grafo Patrón entonces el peso de los ejes no será tenido en cuenta.

Para finalizar mostraremos el formato del archivo con la solución:

```
# Grafo Pesado A y Grafo Patrón B utilizados
E:\Tesis\Grafos\GrafoA1.adg
E:\Tesis\Grafos\GrafoB1.adg
# A B (Mapeos)
1 0
7 1
4 2
# Porcentaje periodo tabu configurado
PorcentajePT=30%
# Penalización por frecuencia configurada
PenalizacionPorFrec=50%
# Frecuencia tolerada configurada
FrecuenciaTolerada=50%
# Cota Superior
Cota=923418.968916
# Estimación iteraciones
Estimacion=819
# Duración del cálculo (HH:MM:SS)
Tiempo=00:00:00.086493
# Sumatoria del peso de los ejes mapeados
Peso=761573.207378
# Periodo TABU
PT=10
# Aspiración por defecto
APD=1
# Aspiración por influencia
API=
# Estadística ( el primero es la solución inicial)
Estadistica=
|-------------------------------------------------------------------------|
| iteración inicial|cant iteraciones |peso obtenido|Tiempo(HH:MM:SS) |
|-------------------------------------------------------------------------|
|                0 |               0 | 332935.6313 | 000:00:00.000053|
|                0 |               1 | 491814.3988 | 000:00:00.000112|
|                1 |               5 | 562451.1719 | 000:00:00.000222|
|                6 |              17 | 566397.6550 | 000:00:00.000332|
|               23 |              52 | 690773.8566 | 000:00:00.000437|
|               75 |              81 | 761573.2074 | 000:00:00.000646|
|-------------------------------------------------------------------------|
```

8 Resultados

8.1 Introducción

En esta sección mostramos los resultados obtenidos por nuestra heurística, basada en la metaheurística Búsqueda Tabú (Tabu Search). Analizaremos en detalle los tiempos de ejecución y la calidad de las soluciones obtenidas.
Para realizar los experimentos, nos vimos en la necesidad de generar instancias de prueba al azar. En consecuencia, desarrollamos un generador de grafos (Apéndice A), el cual, en base a una cantidad de nodos y ejes especificados, construye el grafo asignando peso a los ejes aleatoriamente. También es parametrizable la cantidad de grafos a generar para el número de ejes y nodos solicitado.

Todos los grafos generados tienen pesos en sus ejes y pueden no ser completos, sin embargo, como ya mencionamos en "Implementación", si estuviéramos intentando leer el grafo para utilizarlo como Grafo Pesado se asumirán ejes de peso cero en caso de que dichos ejes no estén especificados en el archivo. Esto es así debido a que nuestro algoritmo trabaja con Grafos Pesados completos.
Si en cambio, estuviéramos intentando leer un Grafo Patrón entonces el peso de los ejes no será tenido en cuenta.

8.2 Pruebas

Para realizar las pruebas se usaron diferentes configuraciones (ver "Archivo de configuración") teniendo en cuenta las siguientes variables:

- Periodo Tabú **(PT)**
- Aspiración por Influencia por Peso Local **(API-PL)**
- Aspiración por Influencia por Peso Global **(API-PG)**
- Aspiración por Influencia por Ejes Local **(API-EL)**
- Aspiración por Influencia por Ejes Global **(API-EG)**
- Sin aspiración por influencia **(API-NA)**
- Frecuencia Tolerada **(FT)** y Penalización por frecuencia **(PF)**
- Aspiración por defecto: menos tabú **(APD-MT)**
- Aspiración por defecto: mejor intercambio **(APD-MI)**
- Aspiración por defecto: sin penalización **(APD-SP)**
- Máxima cantidad de iteraciones **(MCI)**
- Cantidad de iteraciones sin mejora **(ISM)**

A continuación se muestran las diferentes combinaciones de Grafo Patrón y Grafo Pesado que utilizamos durante las pruebas así como las diferentes combinaciones de variables que se aplicaron.

El principal factor que tuvimos que tener en cuenta para la selección de los casos fue un adecuado balance entre la calidad de las pruebas y los tiempos de corridas.

Las pruebas se presentan en tablas donde:

- Los números en las celdas de las tablas indican el porcentaje de densidad de ejes para el Grafo Patrón y densidad de eje con pesos mayores a cero para el Grafo Pesado.
- La fila superior indica la cantidad de nodos del Grafo Patrón.
- La columna de la izquierda indica la cantidad de nodos del Grafo Pesado.

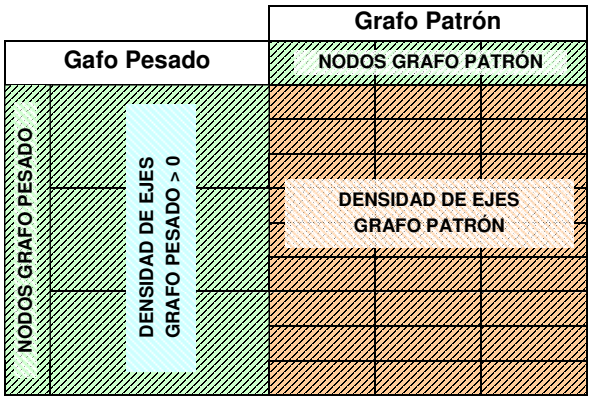


Figura 9 – Esquema de presentación de combinaciones utilizadas

Casos chicos

Estos fueron los únicos casos para los que se corrieron algoritmos exactos.
Se utilizaron Grafos Pesados de 9, 10, 11 y 12 nodos y Grafos Patrones de 3, 4, 5, 6, 7 y 8 nodos. El procesamiento se realizó con las configuraciones detalladas a continuación.

Variable	Valores utilizados
PT	30 60 100
FT	50 20 100
PF	20 50 80
API	PL PG EL EG NA
APD	MT MI SP
MCI	5000
ISM	300

Gafo Pesado		Grafo Patrón 9	6	3
9	100%>0	100%	100%	100%
		80%	80%	66%
		40%	40%	33%
	80%>0	100%	100%	100%
		80%	80%	66%
		40%	40%	33%
	40%>0	100%	100%	100%
		80%	80%	66%
		40%	40%	33%

Gafo Pesado		Grafo Patrón 10	7	4
10	100%>0	100%	100%	100%
		80%	80%	80%
		40%	40%	40%
	80%>0	100%	100%	100%
		80%	80%	80%
		40%	40%	40%
	40%>0	100%	100%	100%
		80%	80%	80%
		40%	40%	40%

Gafo Pesado		Grafo Patrón 11	8	5
11	100%>0	100%	100%	100%
		80%	80%	80%
		40%	40%	40%
	80%>0	100%	100%	100%
		80%	80%	80%
		40%	40%	40%
	40%>0	100%	100%	100%
		80%	80%	80%
		40%	40%	40%

Gafo Pesado		Grafo Patrón 12	8	5
12	100%>0	100%	100%	100%
		80%	80%	80%
		40%	40%	40%
	80%>0	100%	100%	100%
		80%	80%	80%
		40%	40%	40%
	40%>0	100%	100%	100%
		80%	80%	80%
		40%	40%	40%

Figura 10 - Combinaciones utilizadas - Casos chicos

Para cada Grafo Pesado se generaron 3 versiones distintas donde lo único que varía es el peso de sus ejes. Se realizaron todos los procesamientos para cada una de dichas versiones.

Casos medianos

Se utilizaron Grafos Pesados de 20, 30, 40, 50 y 75 nodos y Grafos Patrones de 8, 10, 15, 20, 30, 40, 50 y 75 nodos. El procesamiento se realizó con las configuraciones detalladas a continuación.

Variable	Valores utilizados
PT	30 60 100
FT	50 20 100
PF	20 50 80
API	PL PG EL EG NA
APD	MT
MCI	5000
ISM	800

Gafo Pesado		Grafo Patrón 75	50	20
75	100%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	70%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	40%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	10%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%

Gafo Pesado		Grafo Patrón 50	30	10
50	100%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	70%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	40%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	10%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%

Gafo Pesado		Grafo Patrón 40	30	10
40	100%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	70%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	40%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	10%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%

Gafo Pesado		Grafo Patrón 30	20	10
30	100%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	70%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	40%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	10%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%

Gafo Pesado		Grafo Patrón 20	15	8
20	100%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	70%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	40%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	10%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%

Figura 11 - Combinaciones utilizadas - Casos Medianos

Para cada Grafo Pesado se generaron 3 versiones distintas donde lo único que varía es el peso de sus ejes. Se realizaron todos los procesamientos para cada una de dichas versiones.

Casos grandes

Comprende Grafos Pesados de 100, 250 y 500 nodos y Grafos Patrones de 40, 75, 100, 250 y 500 nodos. En estos casos se utilizaron las siguientes configuraciones:

- Para Grafos Pesados de 100 y 250 nodos: las 6 configuraciones para las cuales se obtuvieron los mejores resultados y las 4 configuraciones para las cuales se obtuvieron los peores resultados durante las pruebas con casos medianos

- Para Grafos Pesados de 500 nodos: la configuración para la cual se obtuvo el mejor resultado y la configuración para la cual se obtuvo el peor resultado durante las pruebas con casos medianos

Grafos de 100 y 250 nodos										
Variable	**Valores utilizados**									
	6 Mejores						**4 Peores**			
PT	30	30	30	30	30	30	100	100	100	100
FT	100	100	100	100	50	20	20	50	50	20
PF	0	0	0	0	80	80	80	80	80	80
API	NA	PL	EG	PG	PG	PG	NA	NA	PG	PG
APD	MT	MT	MT	MT	MT	MT	MT	MT	MT	MT

	100	**250**
MCI	5000	5000
ISM	800	100

Grafos de 500 nodos		
Variable	**Valores utilizados**	
	Mejor	**Peor**
PT	30	100
FT	100	50
PF	0	80
API	PG	NA
APD	MT	MT
MCI	5000	
ISM	100	

Gafo Pesado		Grafo Patrón 100	75	40
100	100%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	70%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	40%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	10%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%

Gafo Pesado		Grafo Patrón 250	100	75
250	100%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	70%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	40%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	10%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%

Gafo Pesado		Grafo Patrón 500	250	100
500	100%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	70%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	40%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	10%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%

Figura 12 - Combinaciones utilizadas - Casos grandes

Para cada Grafo Pesado se generaron 3 versiones distintas donde lo único que varía es el peso de sus ejes. Se realizaron todos los procesamientos para cada una de dichas versiones.

Casos mixtos

Se utilizaron Grafos Pesados de 75, 100, y 250 nodos y Grafos Patrones de 8, 10, 15 y 20 nodos. A diferencia de los casos anteriores, en esta ocasión la cantidad de nodos del Grafo Pesado será significativamente mayor a la cantidad de nodos del Grafo Patrón. El procesamiento se realizó con las configuraciones detalladas a continuación.

Variable	Valores utilizados
PT	30 60 100
FT	50 20 100
PF	20 50 80
API	PL PG EL EG NA
APD	MT
MCI	5000
ISM	800

		Grafo Patrón	
Gafo Pesado		8	10
75	100%>0	100%	100%
		80%	80%
		70%	70%
		10%	10%
	70%>0	100%	100%
		80%	80%
		70%	70%
		10%	10%
	40%>0	100%	100%
		80%	80%
		70%	70%
		10%	10%
	10%>0	100%	100%
		80%	80%
		70%	70%
		10%	10%

		Grafo Patrón		
Gafo Pesado		8	10	15
100	100%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	70%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	40%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	10%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%

		Grafo Patrón		
Gafo Pesado		8	15	20
250	100%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	70%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	40%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%
	10%>0	100%	100%	100%
		80%	80%	80%
		70%	70%	70%
		10%	10%	10%

Figura 13 - Combinaciones utilizadas - Casos mixtos

Para cada Grafo Pesado se generaron 3 versiones distintas donde lo único que varía es el peso de sus ejes. Se realizaron todos los procesamientos para cada una de dichas versiones.

8.3 Análisis de los Resultados

En esta sección realizaremos cuatro tipos de análisis.

1. Análisis de efectividad de la heurística Búsqueda Tabú (TS).
   Llamamos efectividad de un algoritmo a la relación expresada en porcentaje entre el resultado que obtuvo en comparación con un valor de referencia.
   En nuestro caso los algoritmos a evaluar podrían ser la Heurística Inicial (algoritmo goloso que encuentra la solución inicial de TS) o la Heurística Tabú en tanto que el valor de comparación podría ser el valor óptimo o la cota superior.

2. Análisis del porcentaje de mejora que obtiene TS a partir de la solución generada por la heurística constructiva (Heurística Inicial). Se analiza según la configuración utilizada.

3. Análisis del porcentaje de mejora que obtiene TS a partir de la solución generada por la heurística constructiva (Heurística Inicial). Se analiza según la dispersión de los pesos de los ejes del Grafo Pesado.

4. Análisis de los tiempos de corrida.

8.3.1 Efectividad de la Búsqueda Tabú[1]

Con el objetivo de inferir con mayor precisión la efectividad de TS analizamos los resultados obtenidos teniendo en cuenta que sobre grafos chicos, se conoce tanto el valor de la solución óptima como de la cota superior, y sobre grafos grandes, el único elemento de referencia es la cota superior.
El análisis de la Búsqueda Tabú lo hemos dividido en dos partes:

- Primero analizamos los resultados sobre grafos grandes de los cuales no se conoce el resultado exacto.

- Luego analizamos de manera integral los resultados obtenidos sobre grafos chicos en conjunto con la cota superior.
  Esto nos permite comparar la cota superior, el resultado de TS y el resultado exacto pudiéndonos dar una idea de que tan buena es la cota superior y que tan afectada se ve la solución obtenida mediante TS por diferencias entre la cota superior y el resultado Exacto.

[1] En todos los gráficos incluidos en esta sección se representan totales o promedios de todos los tamaños de Grafo Pesado. En el Apéndice B se pueden hallar separados según el tamaño.

Efectividad de la Búsqueda Tabú sobre grafos grandes

Para evaluar la calidad de los resultados utilizamos principalmente la cota superior calculada según el punto *"*Cota superior del peso de la solución*"* puesto que es el único valor de comparación que tenemos disponible para grafos de gran tamaño.

La forma que utilizamos para evaluar la calidad de un resultado es sencilla y se basa en el simple hecho de que la cota es mayor o igual al valor óptimo y de que el algoritmo de búsqueda se acerca por debajo del mismo.

Por consiguiente, la diferencia que exista entre el valor obtenido y la cota superior será la distancia máxima a la que nuestro resultado se encuentra del valor óptimo; ya que de hecho el valor óptimo está entre ambos valores (Figura 14).

Podemos decir entonces que si una solución esta cerca del valor de la cota superior también lo estará del resultado óptimo y que si es igual a la cota superior la solución hallada es el valor óptimo.

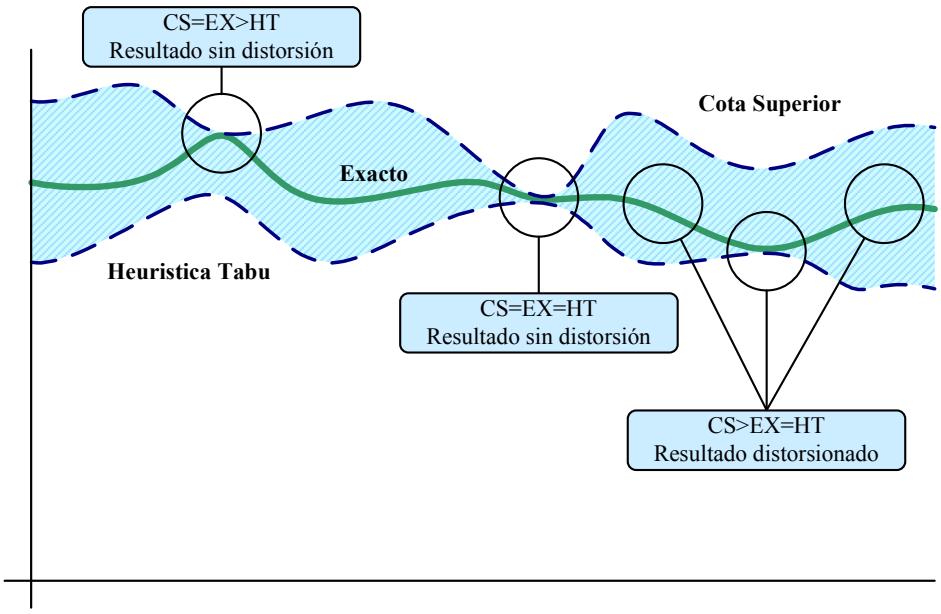


Figura 14 - Relación entre Cota Superior, Exacto y distorsión en la medición de la calidad en los resultados de la Heurística Tabú

El análisis de los resultados contra la cota se puede ver el Figura 15. El mismo agrupa todos los resultados obtenidos para Grafos Pesados que van desde 75 hasta 500 nodos.

Sobre el eje horizontal se observa el grado de efectividad que tuvo el algoritmo de Búsqueda Tabú. Se dividió el grado de efectividad en intervalos de 10% de manera de poder contabilizar cuantos grafos se hallan en cada intervalo.

Sobre el eje vertical se mide la cantidad de casos incluidos en cada rango. Cuantos más casos tengamos cerca del 100% mejor será el desempeño de la heurística.

En este caso y según la Figura 15 podemos decir que en el 25% de los casos el resultado obtenido alcanza el 90% de efectividad y que en un 20% de los casos el resultado obtenido alcanza un 100% de efectividad.

Efectividad de la Heurística Tabú (HT) en comparación con el valor de la cota superior (CS).

$$Efe\_HT\_CS = 100 \times \frac{HT}{CS}$$

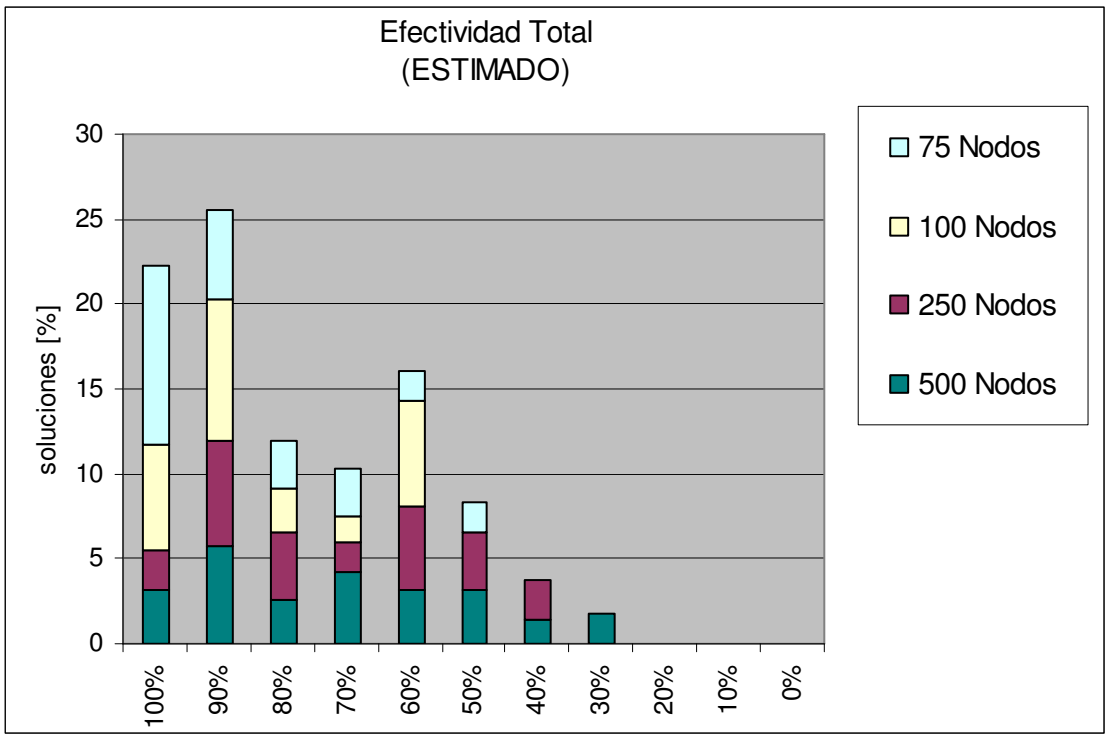


Figura 15 - Efectividad de TS estimada (comparado contra cota) sobre grafos grandes.

Si bien se puede ver que hay una cantidad considerable de casos cuya efectividad es bastante pobre (inferior al 80%), es importante destacar que al encontrarse la cota superior por encima del valor óptimo, la evaluación de la calidad de la solución no es del todo precisa y depende mayormente de la calidad de la cota calculada.

Puesto que hemos demostrado la existencia de casos donde el algoritmo de cálculo de cota tiene un pobre desempeño, decidimos analizar la calidad de la cota. Mediante este análisis intentaremos formar una mejor idea acerca de la eficiencia del algoritmo de búsqueda.

Para tratar de inferir en que medida afecta los resultados la calidad de la cota superior haremos un análisis sobre grafos chicos para los cuales si conocemos el resultado óptimo.

Efectividad de la Búsqueda Tabú y calidad de la cota superior sobre grafos chicos

Para evaluar la calidad de la cota utilizamos los resultados obtenidos mediante algoritmos exactos sobre los grafos más pequeños. Este análisis está representado en la Figura 16. En este caso, el gráfico mide en que medida la cota excede el valor real de la solución. De manera que cuanta más cantidad de casos haya cercanos a 0% mejor será el desempeño del algoritmo que calcula la cota.

Nuevamente se dividió el eje horizontal en intervalos de 10% mientras que el eje vertical muestra el porcentaje de casos que hay contenidos en cada rango.

Exceso de la Cota Superior (ECS) en comparación con el resultado exacto (Ex)

$$ECS = 100 \times \frac{CS - Ex}{Ex}$$

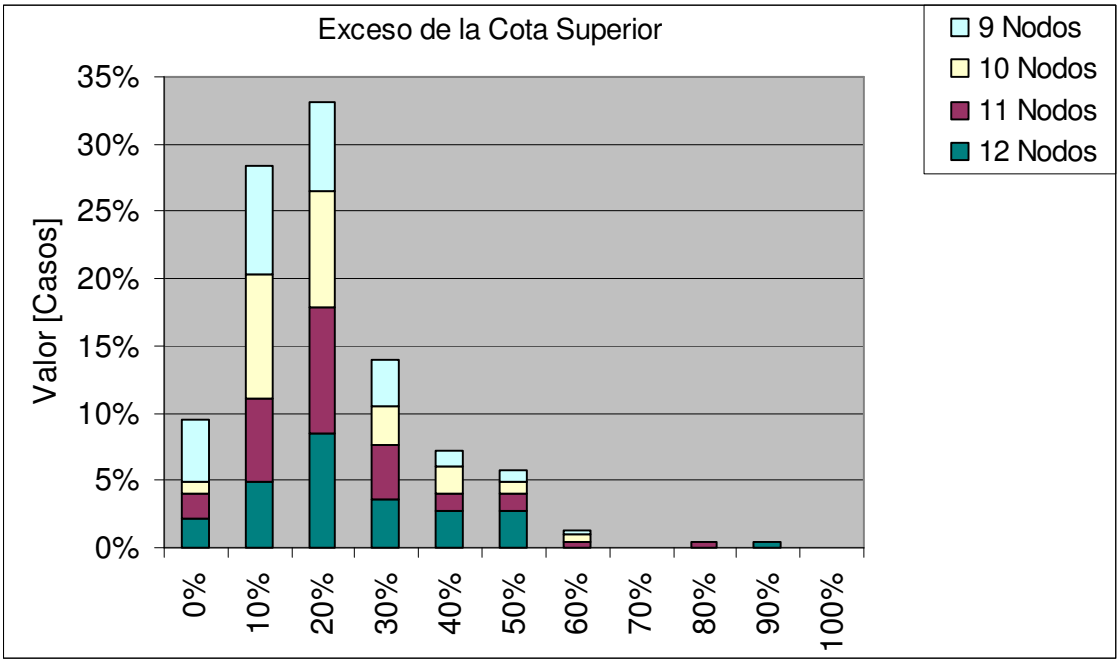


Figura 16 - Exceso del valor de cota calculado sobre grafos chicos comparado contra valor real.

Podemos observar que para una buena cantidad de los casos el valor obtenido para la cota superior fue de mala calidad y que en consecuencia tendrá un gran efecto cuando lo utilicemos para medir la calidad de las soluciones obtenidas mediante la Búsqueda Tabú.
En concreto la relación entre efectividad real (*Efe_HT_Ex*) y efectividad estimada (*Efe_HT_Cs*) se puede expresar en función del exceso de la cota superior (*ECS*):

$$Efe\_HT\_CS = \frac{HT}{CS}$$

… como $CS = (1 + \mathrm{ECS}) \times Ex$ …

$$Efe\_HT\_CS = \frac{HT}{(1 + \mathrm{ECS}) \times Ex}$$

… como $Efe\_HT\_Ex = \frac{HT}{EX}$ …

$$Efe\_HT\_CS = \frac{Efe\_HT\_Ex}{1 + \mathrm{ECS}}$$

Que pasado a porcentaje da:

$$Efe\_HT\_CS = \frac{100}{1 + \mathrm{ECS}} \times Efe\_HT\_Ex$$

Entonces a medida que el exceso de la cota superior toma un valor mayor, la estimación de la efectividad de la heurística se ve afectada proporcionalmente.
En la Figura 17 podemos ver como se vería afectada la efectividad estimada de TS en función del exceso de la cota superior (ECS). Es importante dejar en claro que se graficó hasta un 100% de exceso pero que podría ser aún mayor.

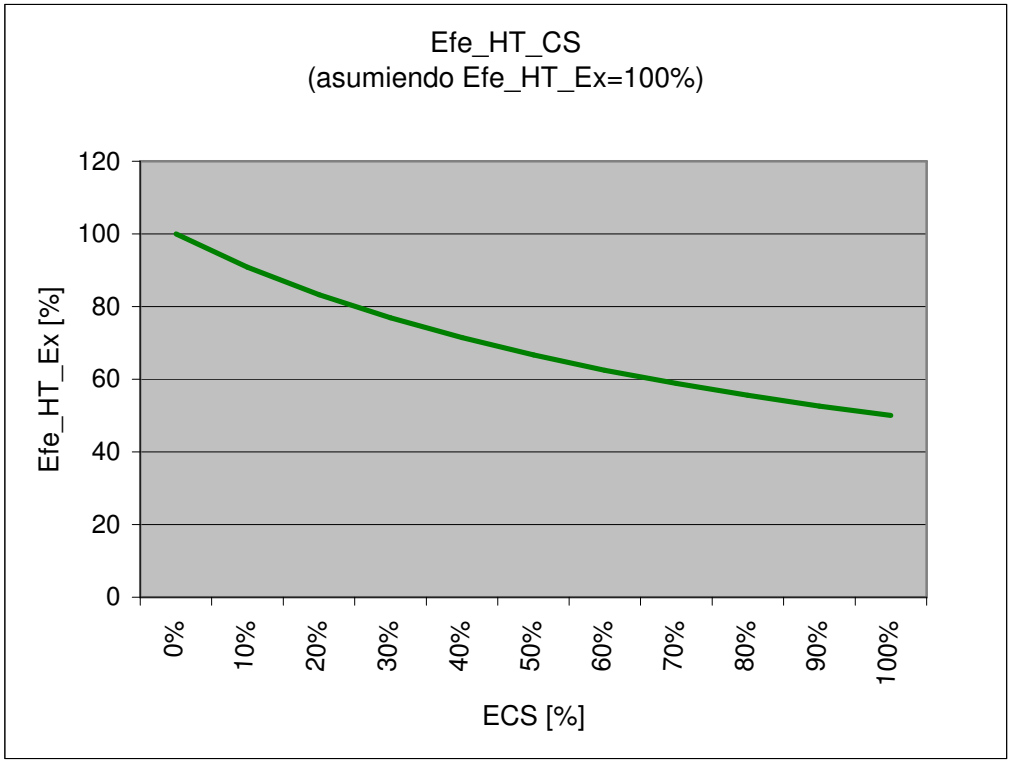


Figura 17 - Deterioro de la efectividad estimada en función del exceso de la cota.

Habiendo formulado en que medida afecta el exceso de la cota superior en la estimación de la calidad de la solución nos centramos en analizar los resultados obtenidos en grafos chicos.
Para ello comparamos el valor de las soluciones de nuestra heurística TS contra dos valores diferentes: la cota superior por un lado y el valor óptimo calculado mediante el algoritmo exacto por el otro.

En la Figura 18 se compara TS contra la cota superior. Se puede observar que la distribución de resultados es similar a la obtenida mediante los grafos grandes y que podemos considerarla aceptable teniendo en cuenta que la mayoría de los resultados estuvieron por encima del 80% de eficiencia.

Sin embargo, cuando en la Figura 19 se realiza la misma comparación en base al valor exacto los resultados cambian bastante. Se observa que en el ciento por ciento de los casos TS alcanzó el valor óptimo.
Esta característica concuerda con el análisis que habíamos realizado sobre la cota superior en la Figura 16. Estando la mayoría de los valores obtenidos para la cota superior entre un 20% y un 30% por encima del valor óptimo es lógico que todas aquellas soluciones que hayan alcanzado un 100% de eficiencia y sean comparadas contra una cota superior excedida se vean afectadas.

Quedando a la vista que los valores estimados de efectividad de TS tienen una distribución similar para grafos chicos y para grafos grandes; y que la efectividad real de los grafos chicos si no se viera afectada por el exceso en la cota superior seria de un cien por ciento para todos los casos:

Podemos suponer que la efectividad estimada de TS está siendo afectada de manera similar para grafos chicos que para grafo grandes. Siendo demasiado optimista pensar que se alcanzó un cien por ciento de efectividad para todos los casos (como sucedió sobre grafos chicos) pero pudiendo asegurar que muchos más casos alcanzaron el valor óptimo de los que se estima mediante la cota superior.

Efectividad de la Heurística Tabú (HT) en comparación con el valor de la cota superior (CS).

$$Efe\_HT\_CS = 100 \times \frac{HT}{CS}$$

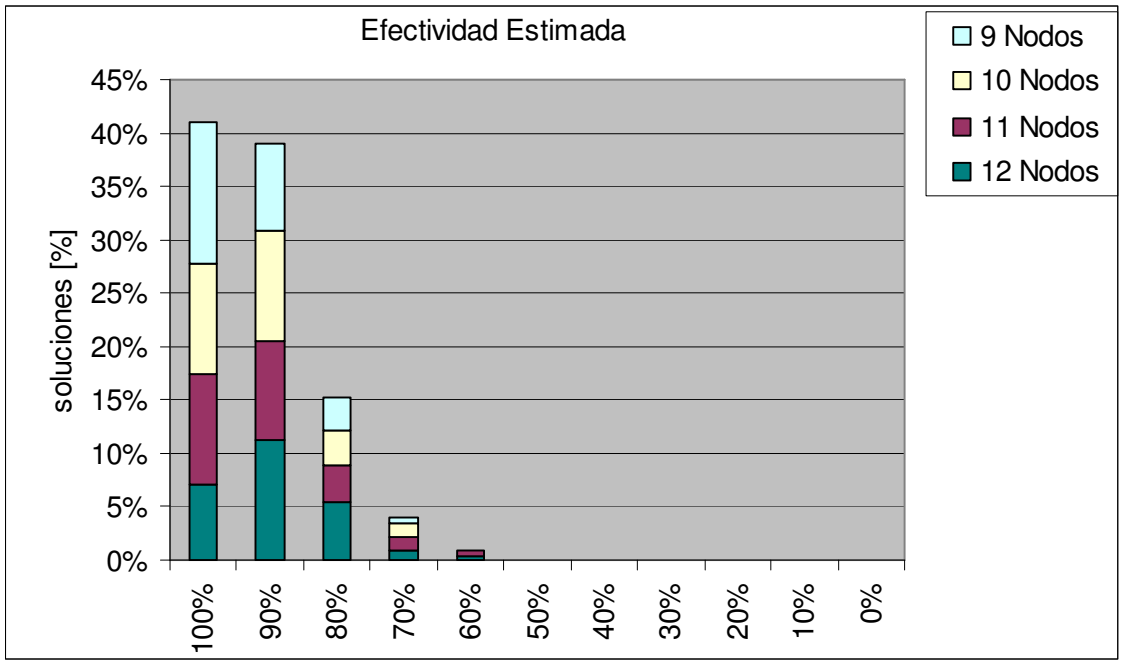


Figura 18 - Efectividad de TS sobre grafos chicos comparado contra cota.

Efectividad de la Heurística Tabú (HT) en comparación con el resultado óptimo obtenido mediante los algoritmos exactos (Ex).

$$Efe\_HT\_Ex = 100 \times \frac{HT}{Ex}$$

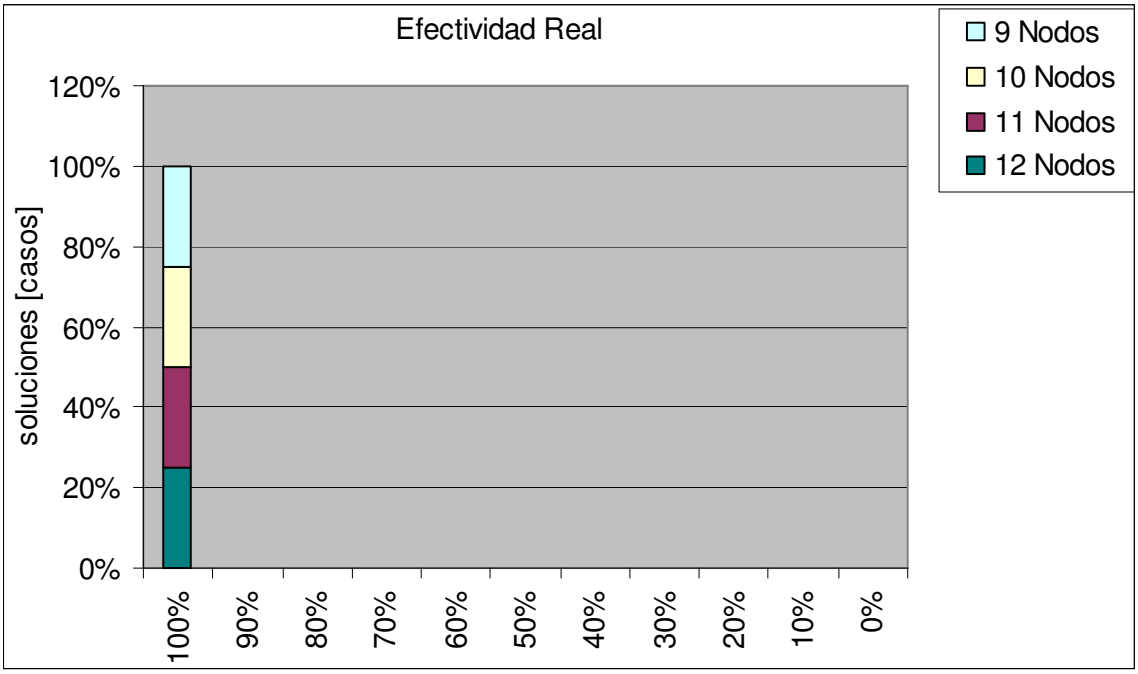


Figura 19 - Efectividad de TS sobre grafos chicos comparado contra valor óptimo real.

8.3.2 Porcentaje de mejora de Tabu Search sobre la heurística inicial

Si bien hemos analizado los resultados obtenidos, dicho análisis nos da una idea del desempeño de la Búsqueda Tabú en conjunto con la Heurística Inicial, pero no nos permite conocer el rendimiento de la Búsqueda Tabú por separado.
Por este motivo ahora nos centraremos en analizar en que porcentaje mejora la Búsqueda Tabú la solución construida mediante la Heurística Inicial.

Por otro lado, la Búsqueda Tabú se puede ejecutar con distintos parámetros de Periodo Tabú, Penalización por Frecuencia, Frecuencia Tolerada, y los distintos tipos de aspiración que afectan el resultado que se obtiene conformando una configuración específica de ejecución. Es por eso que dentro del análisis que realizamos hicimos hincapié en distinguir como varía el porcentaje de mejora en función de la configuración utilizada.

En las Figura 21, Figura 22 y Figura 23 hemos representado los porcentajes de mejora de manera tal que nos permita analizar las distintas configuraciones que hemos probado.

El eje horizontal está dividido según las distintas configuraciones de Periodo Tabú, Frecuencia Tolerada y Penalización por frecuencia. Recordemos que todos los valores se hallan en porcentajes, donde el porcentaje correspondiente al periodo tabú es un valor sobre el máximo periodo tabú tal que siempre haya al menos un movimiento en estado tabú inactivo.

En el eje vertical se observar el porcentaje de mejora correspondiente al promedio de todos los casos tratados mediante cada una de las configuraciones.

En cada gráfico aparecen cinco series correspondientes a los diferentes tipos de aspiración por influencia utilizados: Ejes Global, Ejes Local, Peso Global, Peso Local o sin aspiración por influencia. En "*Aspiración por influencia***"** se explica en que consiste cada uno ellos.

Si bien la cantidad de experimentos que realizamos es mayor, solo incluimos los gráficos correspondientes a tres de los experimentos (9 a 12 Nodos, Hasta 75 Nodos y Grafos Grandes Chicos). El resto corresponde a corridas con grafos más grandes y los resultados fueron idénticos sin importar la configuración utilizada. Adicionalmente a esto, solo se utilizaron seis configuraciones para grafos de 250 nodos y tan solo dos para grafos de 500 nodos por lo que no tenía demasiado sentido graficar los resultados.

Otro aspecto importante para hacer un análisis más preciso del desempeño de la Búsqueda Tabú es tener en cuenta cuanto margen le otorga la Heurística Inicial para mejorar. Es por eso que en este sentido tomamos dos medidas:

Primero hicimos un gráfico análogo a los que realizamos en la sección anterior midiendo la efectividad de la Heurística Inicial. Al realizar el gráfico de efectividad notamos que en una gran cantidad de casos la efectividad era bastante alta.

Al ver estos resultados concluimos que sería adecuado ponderar los gráficos de porcentaje de mejora en función de la efectividad de la Heurística Inicial. Cuando un gráfico es ponderado por cierto valor de efectividad *E,* esto implica que los casos para los cuales la efectividad de la Heurística Inicial fue mayor a *E* no serán tenidos en cuenta para calcular el promedio de efectividad para esa configuración en particular. Teniendo en cuenta que en casi un 50% de los casos la Heurística Inicial alcanzo una efectividad de más del 90% decidimos ponderar los gráficos por este valor.

Efectividad de la Heurística Inicial (HI) en comparación con el valor de la cota superior (CS).

$$Efe\_HI\_CS = 100 \times \frac{HI}{CS}$$

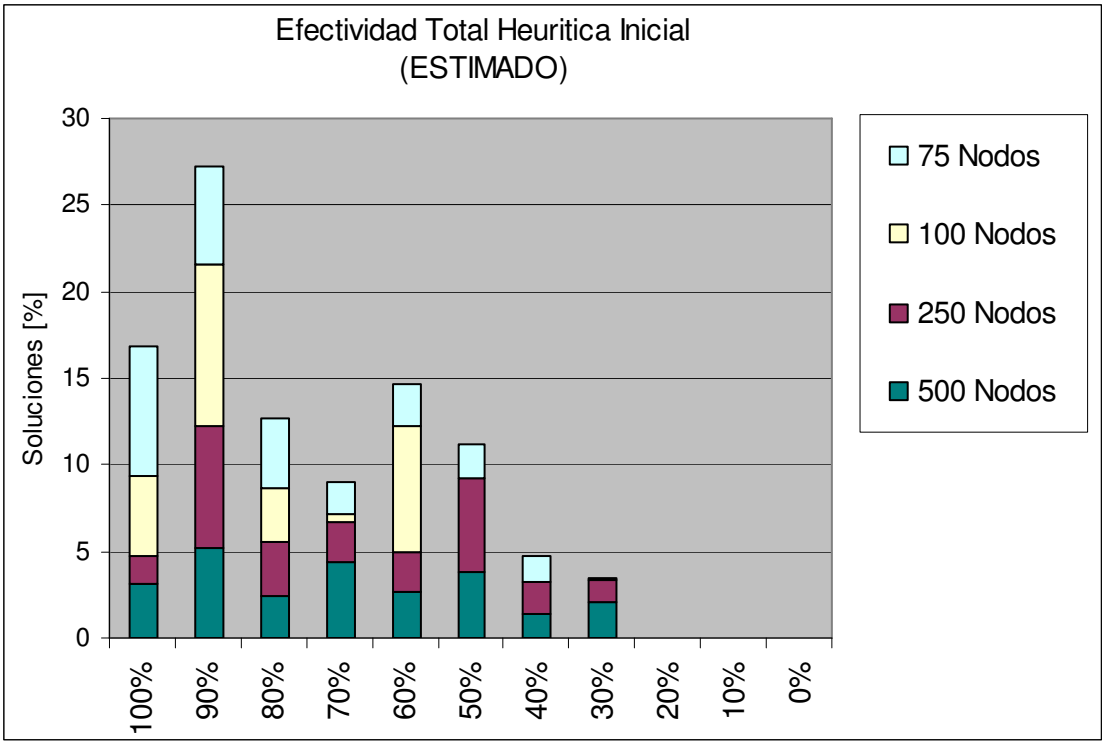


Figura 20 - Efectividad de HI estimada (comparado contra cota) sobre grafos grandes.

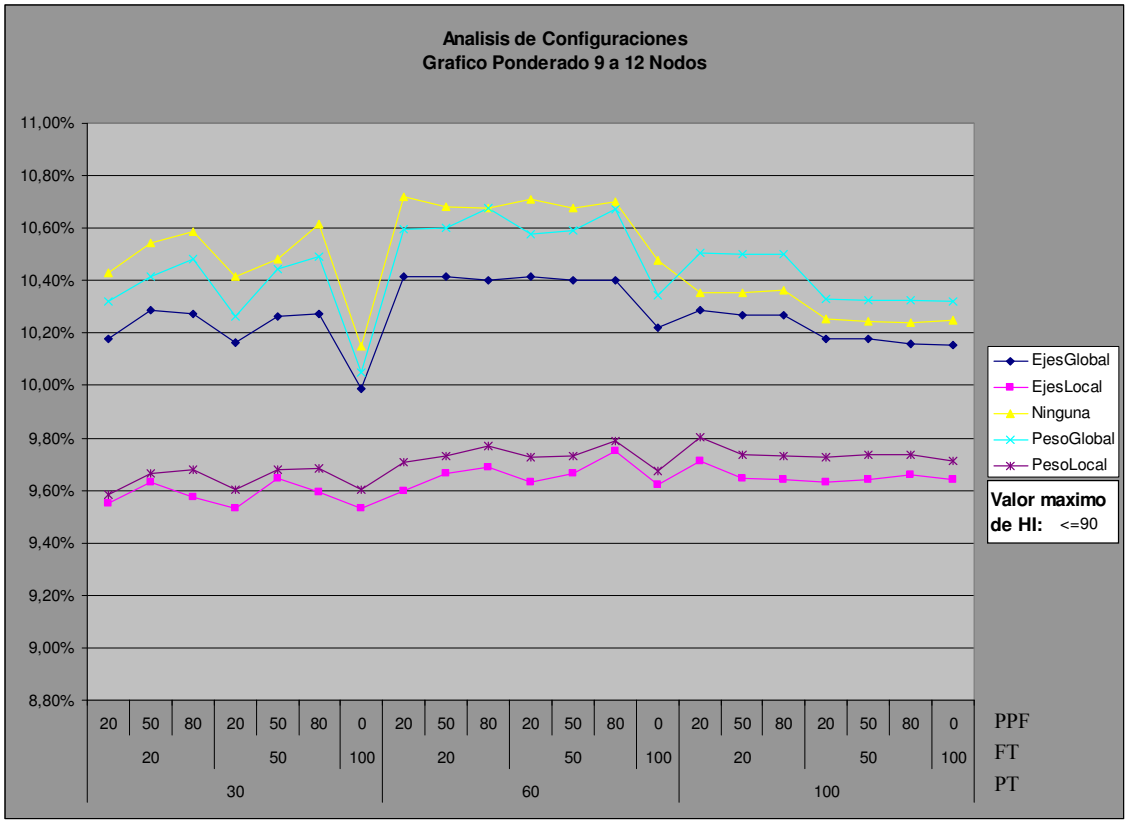


Figura 21 - Análisis de configuraciones – Grafos 9 a 12 Nodos – Casos donde Heurística inicial no alcanzo más del 90 % de efectividad

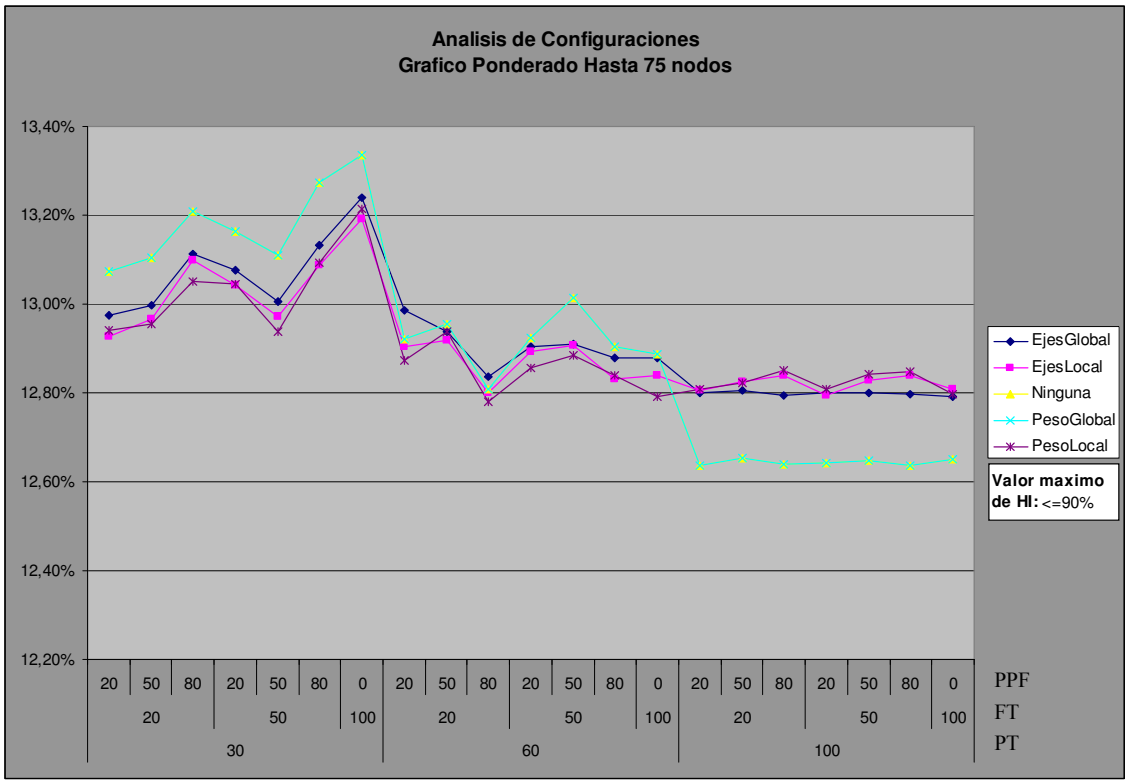


Figura 22 - Análisis de configuraciones – Grafos Hasta 75 Nodos – Casos donde Heurística inicial no alcanzo más del 90 % de efectividad

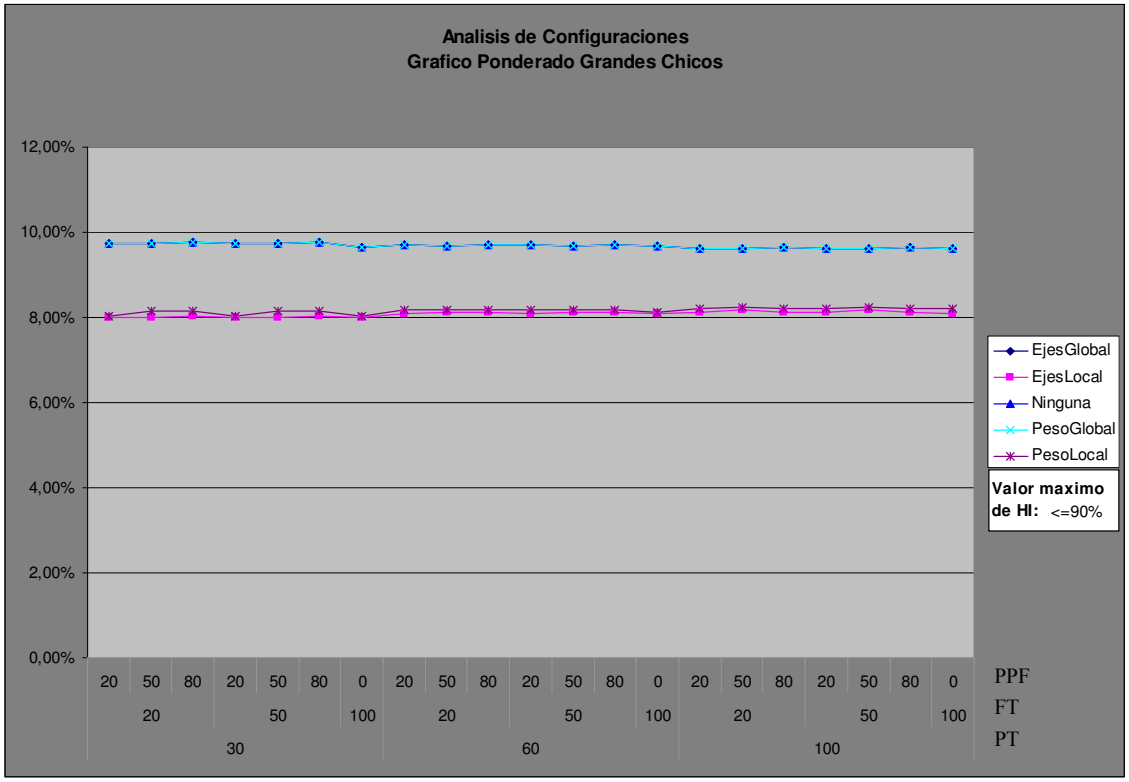


Figura 23 - Grafos grandes contra grafos chicos

En términos generales los mejores resultados se obtuvieron con las configuraciones donde el periodo tabú es más chico (30% y 60%), dando prioridad ligeramente mayor a la exploración local y más intensiva y menor prioridad a la diversificación de la búsqueda.

Por otra parte, salvo en los casos de grafos grandes contra grafos chicos, la aplicación del criterio de aspiración por influencia solo mostró mejoras para las configuraciones con valores altos periodos tabú.

Fue interesante observar que la aspiración por influencia se comportó de manera distinta en grafos grandes y en grafos chicos. Mostrando la aspiración por Peso Global mejoras en los grafos chicos mientras que el resto de las aspiraciones por influencia mostraron mejoras en los casos grandes.

De todos modos, las mejoras obtenidas a través de las aspiraciones por influencia no fueron demasiado significativas y siempre se vieron asociadas a configuraciones donde el periodo tabú era alto.

Creemos que en los casos que la aspiración por influencia mostró resultados fue debido a que compensó los periodos tabú demasiado altos permitiendo elegir intercambios que ante estos valores de periodo tabú se hallaban en estado tabú activo.

En cuanto a la penalización por frecuencia, no hemos encontrado patrones significativos e incluso la tendencia varía de los grafos chicos a los grafos grandes, afectando de manera positiva en los grafos chicos y de manera negativa en los grafos más grandes.

8.3.3 Porcentaje de mejora de Tabu Search sobre la heurística inicial según la dispersión de los pesos del Grafo Patrón

El último análisis que realizamos fue el de la mejora obtenida en función de la dispersión de los pesos de los ejes del Grafo Pesado. La dispersión la calculamos en función de la varianza y la formula que la describe es la siguiente:

$$Disp(A(V_A, E_A, p)), = 100 \times \frac{\sqrt{\frac{1}{|E_A| - 1} \sum_{i=1}^{|E_A|} (p(e_i) - \overline{P(A)})^2}}{\overline{P(A)}}$$

Donde $\overline{P(A)}$ es la media de los pesos de los ejes del Grafo Pesado. Este valor de dispersión ofrece un rango de valores uniforme sin importar el valor absoluto de la solución hallada.

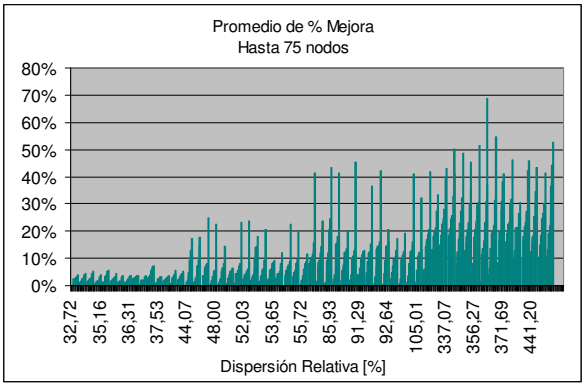
En la Figura 24 se pueden ver los gráficos que confeccionamos para llevar a cabo el análisis. Los mismos están separados según los distintos rangos de tamaños de Grafo Pesado.

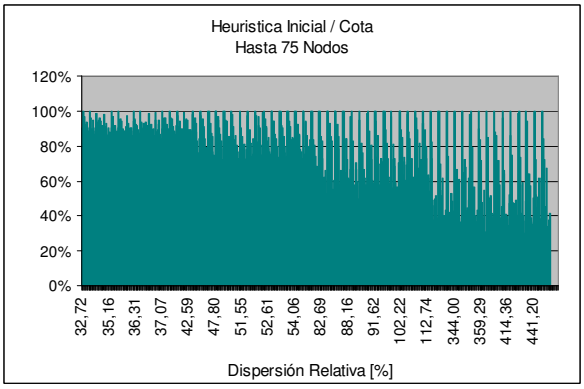
En este caso también se analizaron la heurística inicial y la heurística Tabú de maneras distintas: para la heurística inicial se graficó la "efectividad contra la cota" en el eje vertical mientras que en el eje horizontal se puede ver la dispersión de los pesos; para la heurística Tabú se grafico el porcentaje de mejora sobre el eje vertical y la dispersión de los pesos sobre el eje horizontal.

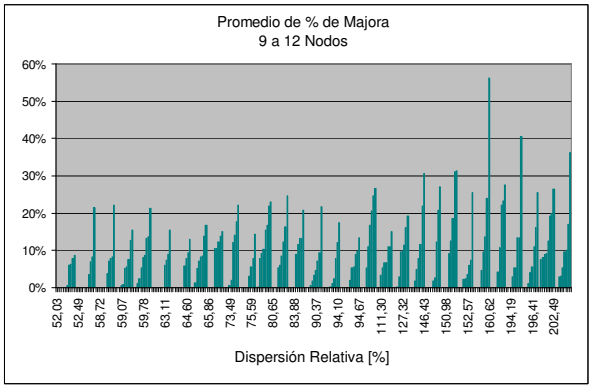
La Figura 24 permite observar que la Búsqueda Tabú incrementa su tendencia a mejorar el resultado obtenido mediante la heurística inicial a medida que los pesos son más dispersos. En contrapartida la heurística inicial, ante la misma situación, tiene una tendencia a obtener peores resultados (soluciones iniciales) y esto puede explicar porque, al tener mayor rango de acción, la Búsqueda Tabú obtiene mayores porcentajes de mejora.

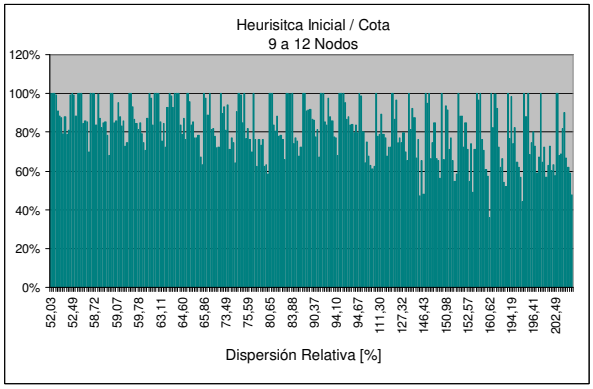
Tabu Search

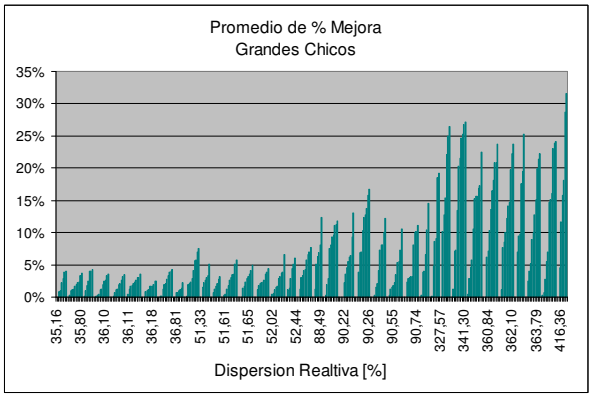
Heurística Inicial

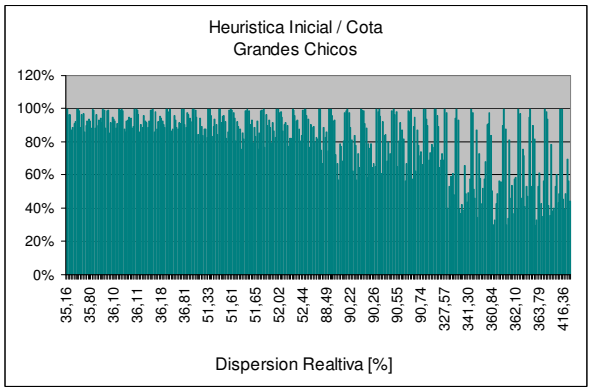


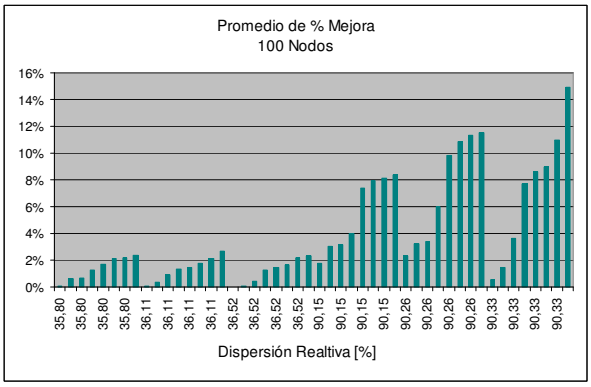












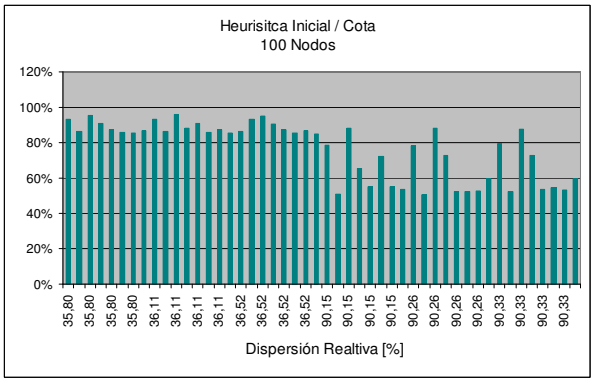


Figura 24 - Efectividad de la heurística inicial / porcentaje de mejora en función de la dispersión de los pesos.

8.3.4 Análisis de tiempos

Siendo la heurística generada mediante la Búsqueda Tabú de tipo polinomial y no conociéndose una solución eficiente para el problema de Isomorfismo de Subgrafos con Peso, no tiene mucho sentido comparar los tiempos de procesamiento de los dos algoritmos.

A modo de ejemplo un caso donde el Grafo Pesado tiene 12 nodos y 40% de cubrimiento de ejes y el Grafo Patrón tiene 8 nodos y un 100% de cubrimiento de ejes (o sea es completo) demoró 8 minutos (489 segundos) mediante el algoritmo exacto mientras que tardó menos de 1 segundo (0.23 segundos) mediante TS.

Demás está decir que muchos de los casos que se incluyeron en las pruebas eran intratables mediante el algoritmo exacto.

En cuanto a los tiempos de procesamiento de la heurística que implementamos, en la Figura 25 se pueden ver los tiempos de procesamiento expresados en segundos. Los tiempos se dividen en tres tipos:

- Tiempo total: el tiempo que demoró el algoritmo en ejecutar la cantidad de iteraciones que se indicó.
- Tiempo HI: el tiempo que demoró la Heurística Inicial en procesar.
- Tiempo TS: el tiempo que demoró la Heurística Tabu Search en hallar la mejor solución (esto no quiere decir que el algoritmo haya finalizado en ese momento).

Se incluyen dos grupos de columnas: uno indica los valores de Tiempo Total, Tiempo HI y Tiempo TS para el caso en que Tiempo TS fue mínimo, el otro indica los mismos valores pero para el caso en que Tiempo TS fue máximo.

	Mínimo			Máximo		
	Tiempo Total (seg.)	Tiempo HI (seg.)	Tiempo TS (seg.)	Tiempo Total (seg.)	Tiempo HI (seg.)	Tiempo TS (seg.)
09 Nodos	0,011538	0,000009	0,000009	0,375629	0,00006	0,131199
10 Nodos	0,013676	0,000015	0,000015	0,151408	0,000052	0,113837
11 Nodos	0,015352	0,000023	0,000023	0,170269	0,000082	0,120834
12 Nodos	0,016688	0,000027	0,000027	0,257532	0,000075	0,218528
Hasta 75 Nodos	0,169705	0,00005	0,00005	108,941838	0,011757	62,978234
Grandes Chicos	0,763302	0,000573	0,000573	184,990867	0,07902	115,700331
100 Nodos	27,34307	0,019447	0,02683	218,374475	0,026539	136,195802
250 Nodos	302,88469	0,176941	0,822311	1891,22054	0,25346	1205,412916
500 Nodos	960,00	0,00	-	92580,00	60,00	-

Figura 25 - Tiempos de procesamiento del algoritmo heurístico.

En la Figura 26 se puede observar el tiempo estimado de procesamiento del algoritmo exacto para los grafos que se incluyeron en las pruebas con la Búsqueda Tabú. El cálculo esta hecho en base a Grafos Patrones de tamaño medio (aproximadamente el

70% de nodos del Grafo Pesado). Se puede ver a simple vista que después de 12 nodos los tiempos se vuelven verdaderamente grandes.

En la Figura 27 se puede ver en detalle el análisis que realizamos para grafos de hasta 20 nodos. Fue mediante este análisis que decidimos realizar pruebas de hasta 12 nodos mediante el algoritmo exacto.

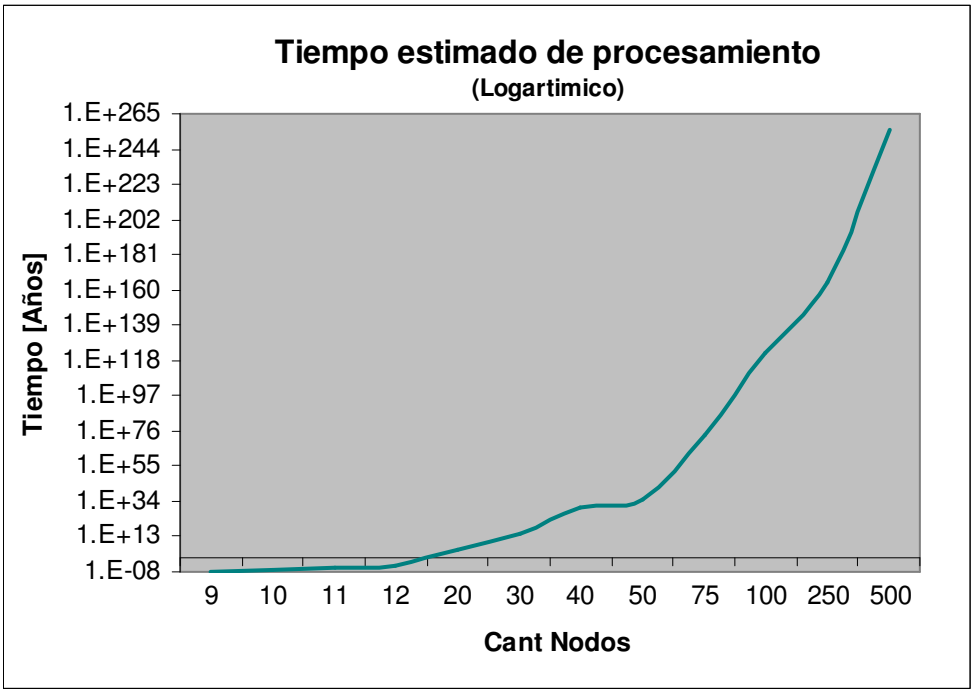


Figura 26 - Tiempo estimado de procesamiento mediante algoritmo exacto (escala logarítmica).

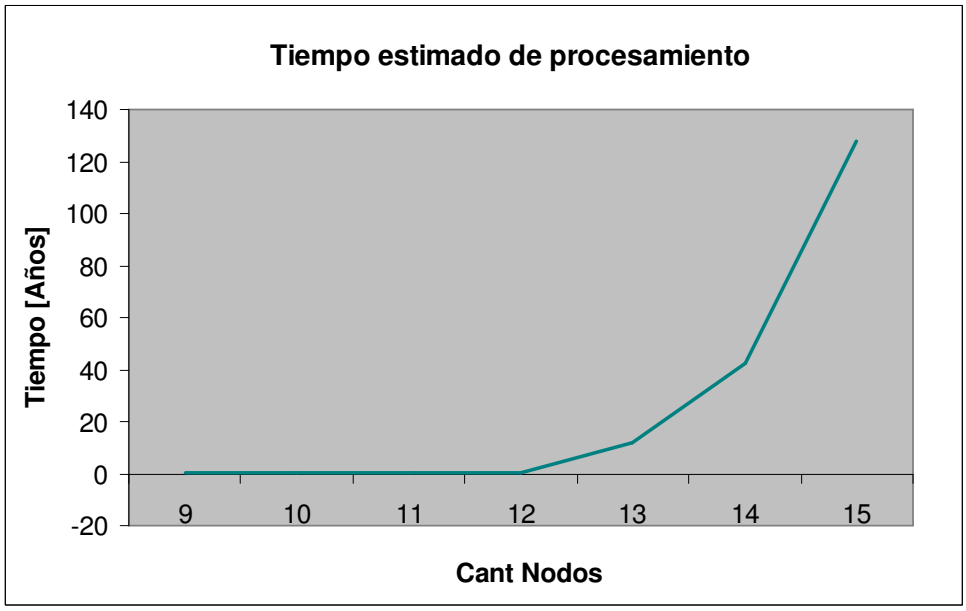


Figura 27 - Tiempo estimado de procesamiento mediante algoritmo exacto.

8.4 Casos especiales

En esta sección realizaremos pruebas con casos grandes los cuales fueron construidos de manera tal que sabemos de antemano el resultado óptimo que deberían alcanzar.
Empezaremos detallando las configuraciones de la Búsqueda Tabú utilizadas en estas pruebas, luego describiremos los casos especiales preparados y por último mostraremos los resultados obtenidos por nuestra heurística.

8.4.1 Configuraciones utilizadas

Al igual que para los casos de Grafos Pesados de 100 y 250 nodos se utilizaron las 6 mejores y las 4 peores configuraciones correspondientes a los casos de Grafos Pesados de 75 nodos:

Casos con grafos grandes y resultado conocido										
Variable	**Valores utilizados**									
	6 Mejores						**4 Peores**			
PT	30	30	30	30	30	30	100	100	100	100
FT	100	100	100	100	50	20	20	50	50	20
PF	0	0	0	0	80	80	80	80	80	80
API	NA	PL	EG	PG	PG	PG	NA	NA	PG	PG
APD	MT	MT	MT	MT	MT	MT	MT	MT	MT	MT

8.4.2 Caso 1 – "Multi K_{10}"

Como podemos ver en la Figura 28, el Grafo Patrón utilizado para este caso es un grafo completo de 10 nodos (**K_{10}**).
El Grafo Pesado (Figura 29) está compuesto por 9 subgrafos **K_{10}** unidos todos a través de un mismo nodo. El peso de los ejes dentro de cada **K_{10}** es 9, en cambio el peso de los ejes que unen estos subgrafos es 10.

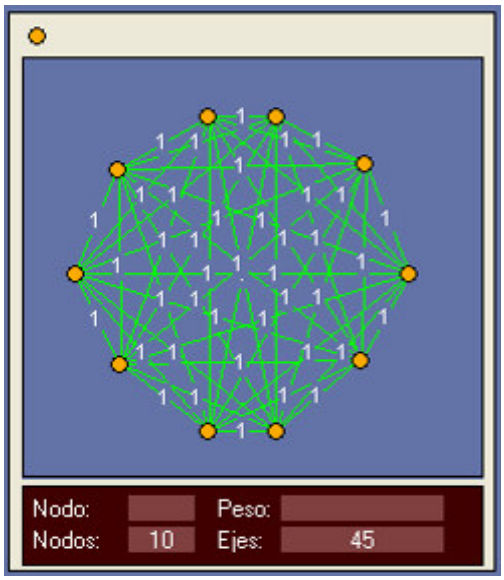


Figura 28 – Grafo Patrón – "Multi K_{10}"

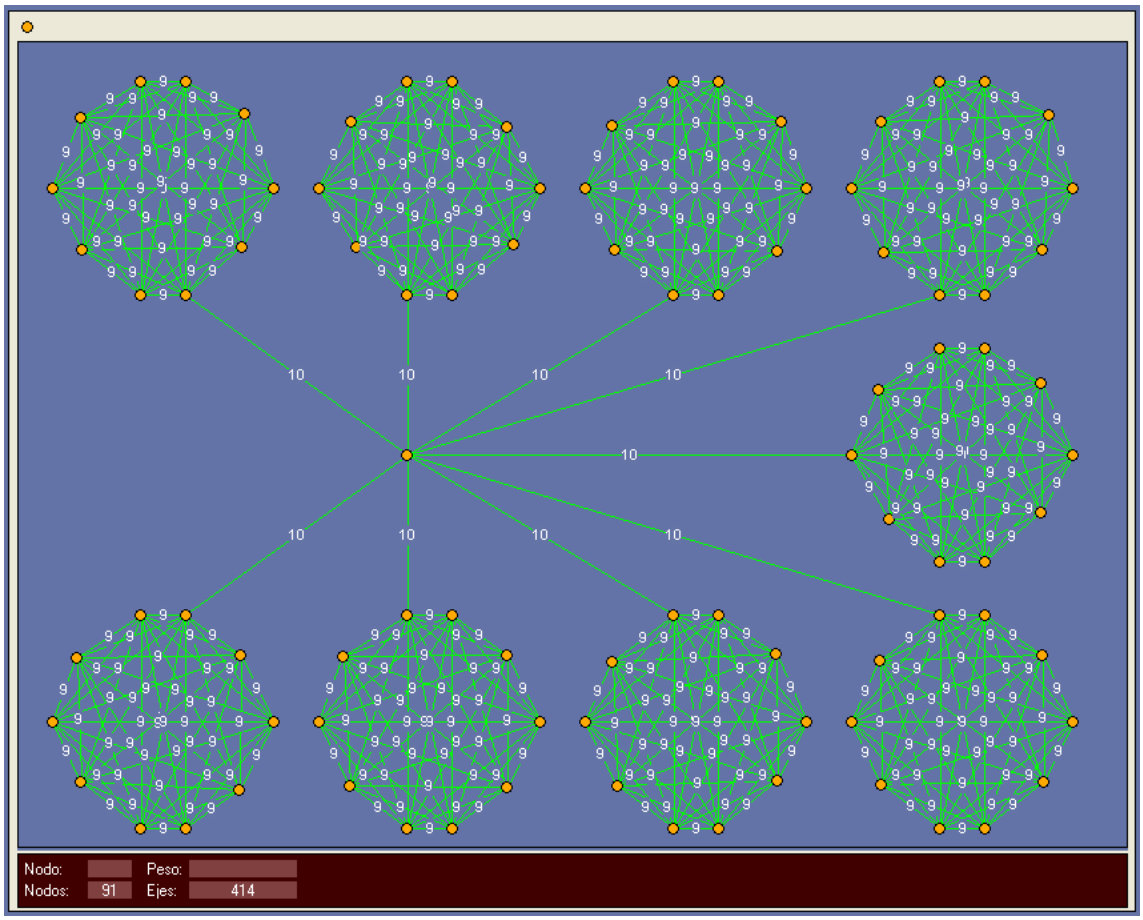


Figura 29 - Grafo Pesado - "Multi K_{10}"

Observando la estructura de ambos grafos podemos notar que el Grafo Patrón debe mapearse contra cualquiera de los subgrafos $\mathbf{K_{10}}$ del Grafo Pesado para obtener así la solución óptima.
De esta manera el peso de dicha solución debería ser:
9 x #ejes del $\mathbf{K_{10}}$ = 9 x 45 = 405

El hecho de que dentro de cada $\mathbf{K_{10}}$ el peso de los ejes sea 9 pero que los ejes utilizados para unir dichos $\mathbf{K_{10}}$ entre sí tengan peso 10, se hizo especialmente para que la heurística que busca una solución inicial del tabú no encuentre la solución óptima y delegue entonces esa tarea a la Búsqueda Tabú pura.
Podemos ver como realizaría el primer mapeo la heurística inicial siguiendo los 4 pasos iniciales de su algoritmo:

A = Grafo Pesado
B = Grafo patrón

1) Se toma el nodo de mayor grado de ***B***.
   En caso de haber más de uno, se elige uno al azar.

2) Δ_B = grado del nodo obtenido en (1).

3) Se toma el nodo de ***A*** donde la suma de los Δ_B ejes más pesados relacionados a dicho nodo sea la mayor.
   En caso de haber más de uno, se elige uno al azar.

4) Mapeo el nodo obtenido en (1) con el obtenido en (3).

En el paso 1 tomamos cualquier nodo ya que como el Grafo Patrón es un $\mathbf{K_{10}}$ todos los nodos tienen grado 9. Luego en el paso 3, tomamos el nodo del Grafo Pesado donde la suma de sus 9 ejes más pesados sea máxima. De esta manera, mapearemos el primer nodo del Grafo Patrón contra el nodo del Grafo Pesado utilizado para unir a los 9 subgrafos $\mathbf{K_{10}}$ que conforman al mismo. Por la estructura de ambos grafos y por el valor de los ejes del Grafo Pesado sabemos que dicho nodo no forma parte de la solución óptima con la cual dejaremos en manos de la Búsqueda Tabú pura el arreglo de este mapeo "erróneo" de la heurística inicial.

8.4.3 Caso 2 – "Multi Ciclo"

En este caso nuestro Grafo Patrón (Figura 30) es un ciclo de 9 nodos.
El Grafo Pesado (Figura 31) está compuesto por varios ciclos de 9 nodos unidos entre sí a través de un mismo nodo. El peso de los ejes dentro de cada ciclo es 9, en cambio el peso de los ejes que unen dichos ciclos es 10.

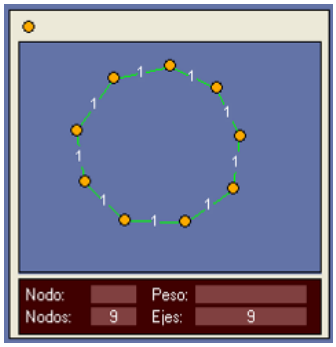


Figura 30 - Grafo Patrón - "Multi Ciclo"

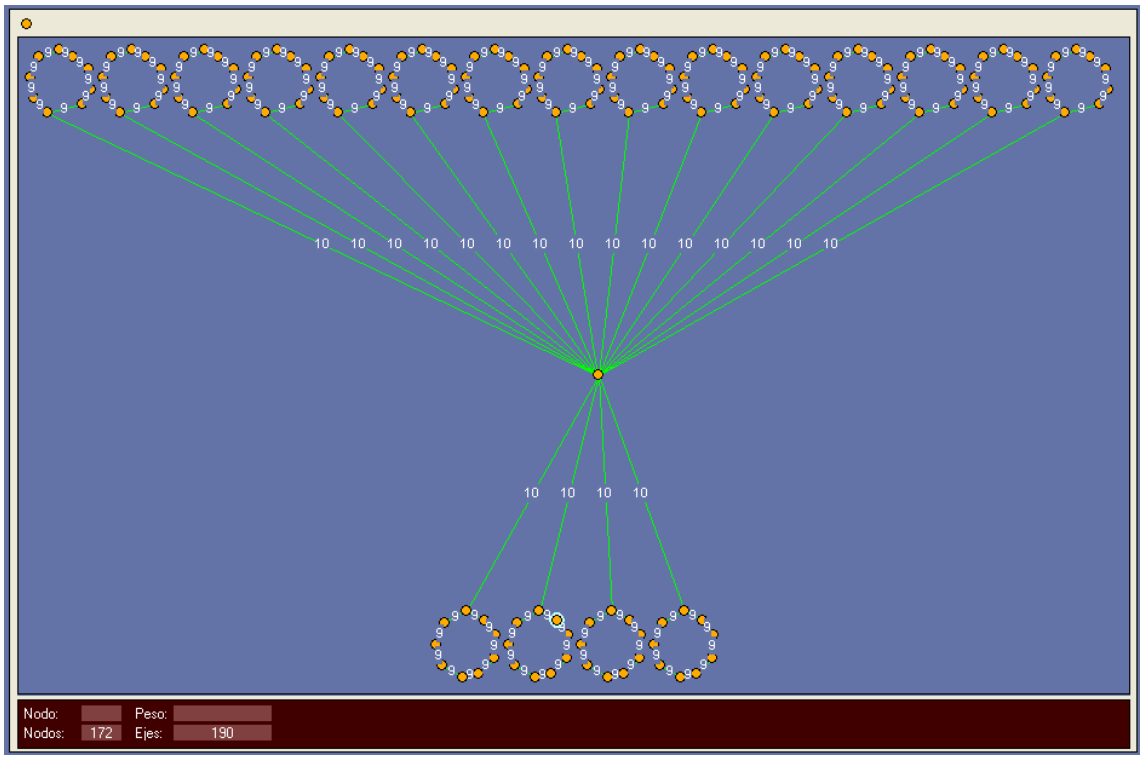


Figura 31 - Grafo Pesado - "Multi Ciclo"

Observando la estructura de ambos grafos podemos notar que el Grafo Patrón debe mapearse contra cualquiera de los ciclos del Grafo Pesado para obtener así la solución óptima.
De esta manera el peso de dicha solución debería ser:
9 x #ejes del ciclo = 9 x 9 = 81

Al igual que en el caso anterior, el hecho de que el peso de los ejes que unen los ciclos sea 10 y que el peso de los ejes que componen los ciclos sea 9, es simplemente para que la heurística inicial de la Búsqueda Tabú no encuentre la solución óptima.

8.4.4 Caso 3 – “Rueda de bicicleta”

En este caso nuestro Grafo Patrón (Figura 32) es una mezcla entre un ciclo y una estrella, lo cual nos deja una forma de “rueda de bicicleta o carreta”. Es un ciclo de 73 nodos con el agregado de un 74º nodo que se relaciona con cada uno de los nodos del ciclo.
El Grafo Pesado (Figura 33) está compuesto por un subgrafo similar al Grafo Patrón unido a una estrella de 74 nodos, o sea es la unión de la “rueda de bicicleta” con una “estrella”. El peso de los ejes siempre es 1 a excepción de dos ejes pertenecientes a la estrella los cuales tienen peso 2.

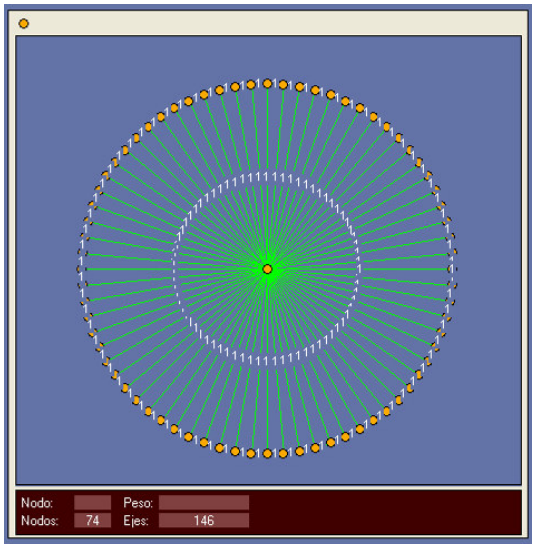


Figura 32 - Grafo Patrón - “Rueda de bicicleta”

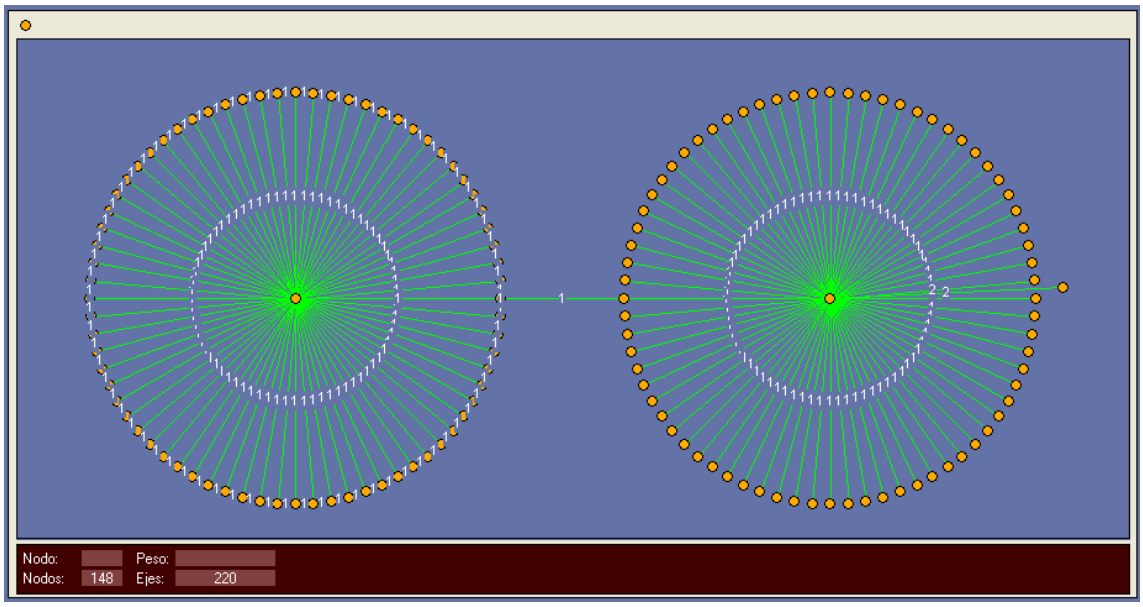


Figura 33 - Grafo Pesado - “Rueda de bicicleta”

Observando la estructura de ambos grafos podemos notar que el Grafo Patrón debe mapearse contra el subgrafo con forma de "rueda de bicicleta" del Grafo Pesado.
De esta manera el peso de dicha solución debería ser:
(#ejes del ciclo + #ejes de la estrella) x peso = (73 + 73) x 1 = 146

De la misma manera que en los casos anteriores, el hecho de que el peso de 2 de los ejes de la estrella sea igual a 2 y que el resto del peso de los ejes que componen al Grafo Pesado sea igual a 1, es simplemente para que la heurística inicial de la Búsqueda Tabú no encuentre la solución óptima.

8.4.5 Resultados

En los 3 casos especiales analizados el resultado alcanzado por la Búsqueda Tabú fue el óptimo para todas las configuraciones. La única diferencia radicó en la cantidad de iteraciones que necesitó el algoritmo para llegar a dicho resultado.
También pudimos observar que la cota encontrada para estos casos estuvo muy cerca del óptimo.

El siguiente cuadro muestra los resultados alcanzados por todas las configuraciones:

Caso	Resultado óptimo	Cota	Heurística inicial	Búsqueda Tabú pura	Porcentaje de mejora
Multi K_{10}	405	414	90	405	77,78%
Multi Ciclo	81	86	74	81	8,64%
Rueda de bicicleta	146	149	74	146	49,32

9 Conclusiones

El trabajo consistió en la creación de una heurística basada en la metaheurística Búsqueda Tabú para la resolución del problema de Isomorfismo de Subgrafos con Pesos.

Con el objetivo de determinar la calidad de los resultados hallamos una cota superior sencilla de calcular que nos permite estimar, dependiendo de que tan buena sea dicha cota, la calidad de las soluciones encontradas por la heurística.

Pudimos observar que el rendimiento del algoritmo se halla entre un cincuenta y un noventa por ciento del valor de la cota superior calculada. Además vimos, al comparar contra casos chicos de los cuales conocíamos el valor óptimo, que la evaluación de la calidad de los resultados estaba siendo fuertemente afectada por la mala calidad de la cota en ciertos casos.
Sobre este punto podría intentarse hallar un algoritmo que mejore la cota calculada.

En cuanto a los casos especiales analizados, los cuales fueron construidos de manera tal que sabemos de antemano el resultado óptimo que deberían alcanzar, vimos que nuestro algoritmo encontró dicha solución óptima en todas las ocasiones. Esto refuerza la idea anterior de que la evaluación de la calidad de los resultados podría ser mejorada en un trabajo futuro.

Utilizamos tres criterios de aspiración por defecto y notamos que esta variable no afecta en el resultado final. Si bien es necesario tener un criterio de aspiración por defecto, es indistinto cual de los tres se utiliza.

Probamos cuatro criterios de aspiración por influencia basándonos en el cálculo de la influencia según los elementos locales a la solución o de manera global dentro del grafo. Los resultados se vieron afectados en diferente forma a medida que crecía el tamaño de los grafos utilizados pero en muy pocos casos los criterios de aspiración por influencia mejoraron los resultados.

Para las pruebas con grafos chicos (menos de 12 nodos), los mejores resultados los obtuvimos cuando utilizamos influencia basada en elementos globales o no utilizamos aspiración por influencia.

En cambio en los casos medianos (grafos de entre 20 y 75 nodos), observamos que la influencia calculada en base a elementos locales era la que proveía mejoras.

En todos los casos la aspiración por influencia solo fue efectiva cuando el periodo tabú es máximo, o sea, cuando tenemos más intercambios en estado tabú activo, con lo cual hay más candidatos a aspirar por influencia.

Notamos luego que a medida que crecían en cantidad de nodos los grafos (100 nodos o más), los resultados obtenidos se parecían sin importar el criterio de aspiración por influencia aplicado, o si utilizábamos o no aspiración por influencia.
Un punto interesante a tratar en trabajos futuros sería la búsqueda de algoritmos de cálculo de influencia que permitan hallar mejoras apreciables sobre el rendimiento de la heurística.

En el análisis por dispersión, observamos que la Búsqueda Tabú tiende a mejorar su rendimiento a medida que los pesos del Grafo Pesado se hacen más dispersos pero también disminuye el rendimiento de la heurística inicial dándole más margen a la Búsqueda Tabú para encontrar mejoras.
Sería interesante experimentar con otros algoritmos de búsqueda de solución inicial que no se vean afectados por la dispersión.

Nos resulto interesante comprobar que mediante el uso de la Búsqueda Tabú, y luego de pocas iteraciones, ya se hallan buenos resultados. Las sucesivas mejoras demoran en encontrarse y no suelen ser sustanciales, con lo cual vale la pena analizar otro criterio de parada de la Búsqueda Tabú como por ejemplo un corte según la cercanía de la cota con la solución encontrada.

Como mejora o variante posible a nuestra implementación de la heurística basada en la metaheurística Búsqueda Tabú, también podemos pensar en modificar el entorno de solución con el que trabajamos. En nuestro caso, dada la simpleza con la que obtenemos todos los intercambios posibles y sus respectivos valores de intercambio, no realizamos podas. Por esa razón, un nodo perteneciente a la solución, posee como vecinos a todos los restantes nodos del Grafo Pesado.

Por último, sería interesante como trabajo futuro, obtener la resolución del problema Isomorfismo de Subgrafos con Pesos basando la implementación en otras metaheurísticas distintas a la Búsqueda Tabú, de manera de poder comparar los resultados obtenidos.
También sería interesante idear otras transformaciones de instancias que permitan minimizar la distorsión que se obtiene sobre la efectividad de la heurística.

10 Bibliografía

[1] D. Applegate, R. E. Bixby, V. Chvátal, W. J. Cook “Concorde” (TSP – Aplicación) http://www.tsp.gatech.edu/concorde.html

[2] C. A. Coello, “Búsqueda Tabú: Evitando lo Prohibido”, Soluciones Avanzadas. Tecnologías de Información y Estrategias de Negocios, Año 5, Número 49, pp. 72-80 (15 de septiembre de 1997)

[3] K. A. Dowsland, B. Adenso Díaz, “Diseño de Heurísticas y Fundamentos del Recocido Simulado”, Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.20 (2001),pp. 34-52 ISSN: 1137-3601. © AEPIA http://www.aepia.dsic.upv.es

[4] M. R. Garey and D. S. Johnson, “Computers and Intractability: A Guide to the Theory of NP-completeness”, W.H. Freeman and Company (1979).

[5] F. Glover, "Tabu Search, Part I", ORSA Journal on Computing, 1:3, pp. 190-206. (1989)

[6] F. Glover, "Tabu Search, Part II", ORSA Journal on Computing, 2:1, pp. 4-31. (1990a)

[7] F. Glover, "Tabu Search: A Tutorial", Interfaces, Vol 20, No. 4, pp. 74-94. (July-August 1990)

[8] Glover, F. and M. Laguna , “Tabu Search”, Kluwer Academic Publishers, Boston (1997)

[9] F. Glover, B. Melián, “Tabu Search”. ,Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No. 19 (2003), pp. 29-48. ISSN: 1197-3601© AEPIA http://www.aepia.org/revista

[10] P. Hansen, N. Mladenovic, J. A. Moreno Pérez, “Variable Neighbourhood Search”, Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.19 (2003),pp. 77-92, ISSN: 1137-3601. © AEPIA (http://www.aepia.org/revista).

[11] F. Harary, “Graph Theory”, Addison–Wesley, Reading, MA. (1969)

[12] R. Martí, M. Laguna, “Scatter Search”, Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.19 (2003),pp. 123-130, ISSN: 1137-3601. © AEPIA (http://www.aepia.org/revista).

[13] R. Martí , J. M. Moreno Vega, “MultiStart Methods”, Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.19 (2003),pp. 49-60, ISSN: 1137-3601. © AEPIA (http://www.aepia.org/revista).

[14] B. Melián, J. A. Moreno Pérez, J. M. Moreno Vega, “Metaheuristics: A global view”, Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.19 (2003),pp. 7-28, ISSN: 1137-3601. © AEPIA (http://www.aepia.org/revista).

[15] P. Moscato, C. Cotta, “An Introduction to Memetic Algorithms”, Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.19 (2003),pp. 131-148 ISSN: 1137-3601. © AEPIA (http://www.aepia.org/revista).

[16] M. G. Resende, J. L. González Velarde, “GRASP: Greedy Randomized Adaptive Search Procedures”, Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.19 (2003),pp. 61-76, ISSN: 1137-3601. © AEPIA (http://www.aepia.org/revista).

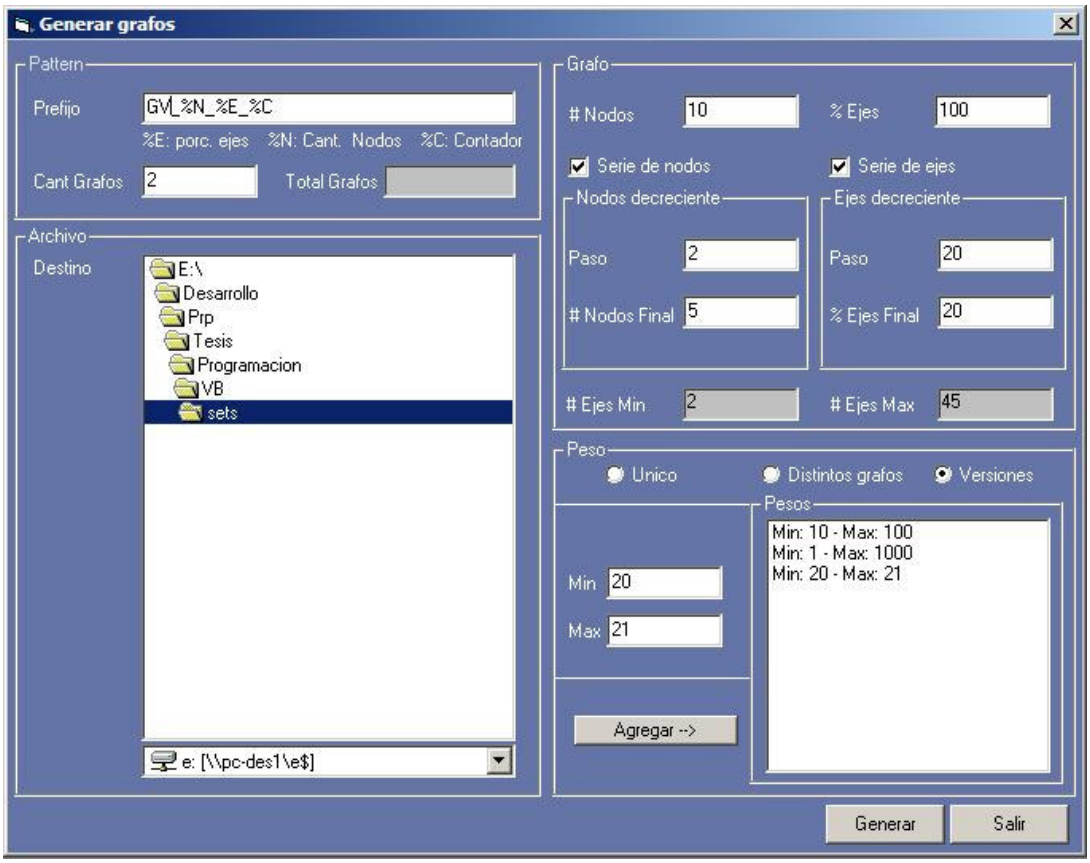
[17] A. Reum Lee, “Identificación de detalles Estructurales en proteínas mediante el método de búsqueda de subgrafos isomorfos frecuentes”, Tesis de Licenciatura, Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires (Mayo de 2008).

11 Apéndice A

Con el objetivo de utilizar grafos de un tamaño considerable para nuestras pruebas nos vimos en la necesidad de realizar un software para el armado de grafos.

11.1 Generador de grafos

El software de generación de grafos posee la capacidad de ser parametrizado con el fin de armar un conjunto de grafos con características similares.
Asimismo, cabe la posibilidad de generar series de estos conjuntos de grafos donde especificamos la cantidad de nodos y el porcentaje de ejes que tendrán los grafos del conjunto inicial, para luego indicar como irán variando estas dimensiones hasta llegar a un conjunto final.
Además, se puede indicar cuantos grafos se desea generar en cada conjunto.
Por último, se pueden suministrar varios juegos de valores para los pesos máximo / mínimo de manera que se generen para cada conjunto uno o más grafos (dependiendo del parámetro de cantidad de grafos) con pesos en los ejes que oscilen dentro de dichos valores.
Con el objetivo de diferenciar cada uno de los archivos de grafos generados el programa permite ingresar un pattern para el nombre de archivo con comodines para el porcentaje de ejes (%E), la cantidad de nodos (%N) y un contador para los casos en que sea necesario diferenciar varios grafos con igual cantidad de nodos y porcentaje de ejes.



11.2 Algoritmo de generación de grafos

El algoritmo de generación de grafos consiste en seleccionar del conjunto total de ejes numerados en cierto orden un subconjunto del tamaño indicado por el usuario para luego asignar a este subconjunto sus pesos.
El armado de un grafo comienza generando una lista de todos los posibles ejes que podrían componerlo, para luego seleccionar aleatoriamente, en una especie de sorteo, los ejes que finalmente conformarán el grafo hasta alcanzar la cantidad de ejes indicada por el usuario. A medida que se seleccionan los ejes van siendo marcados para evitar seleccionarlos más de una vez. De esta manera en cada sucesiva iteración el "sorteo" es realizado entre los ejes que aun no están marcados y así se evita tener que verificar que el numero de eje seleccionado sea válido.
Veamos un ejemplo para un grafo de 5 nodos con un porcentaje de cubrimiento del 50% de los ejes:

Nodos: 5
Total de ejes: 10
Ejes a completar: 5
Lista de ejes:

1	2	3	4	5	6	7	8	9	10
(1,2)	(1,3)	(1,4)	(1,5)	(2,3)	(2,4)	(2,5)	(3,4)	(3,5)	(4,5)

1) Eje seleccionado: 1 (entre 10)

1	2	3	4	5	6	7	8	9	10

2) Eje seleccionado: 1 (entre 9)

1	2	3	4	5	6	7	8	9	10

3) Eje seleccionado: 5 (entre 8)

1	2	3	4	5	6	7	8	9	10

4) Eje seleccionado: 7 (entre 7)

1	2	3	4	5	6	7	8	9	10

5) Eje seleccionado: 3 (entre 6)

1	2	3	4	5	6	7	8	9	10

Lista de ejes seleccionados

1	2	3	4	5	6	7	8	9	10
(1,2)	(1,3)	(1,4)	(1,5)	(2,3)	(2,4)	(2,5)	(3,4)	(3,5)	(4,5)

Una vez obtenida la lista de ejes seleccionados resta asignar los pesos de manera aleatoria.

12 Apéndice B

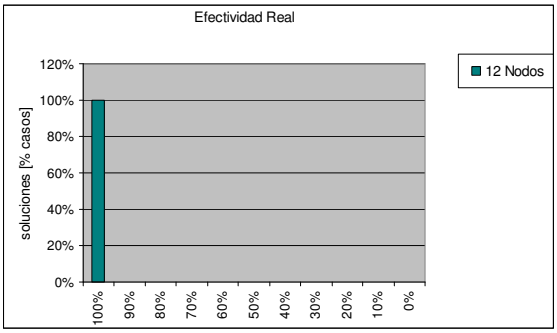
A continuación se incluyen los gráficos de resultados detallados por tamaño de Grafo Pesado.

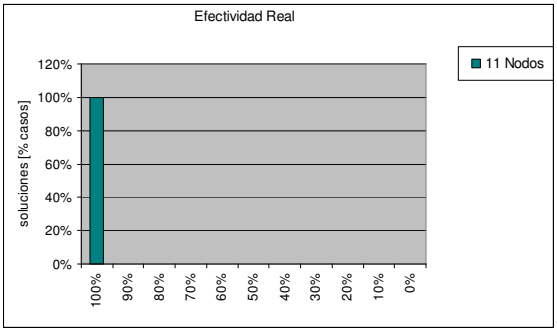
12.1 Efectividad de la heurística en Grafos de 9 a 12 nodos

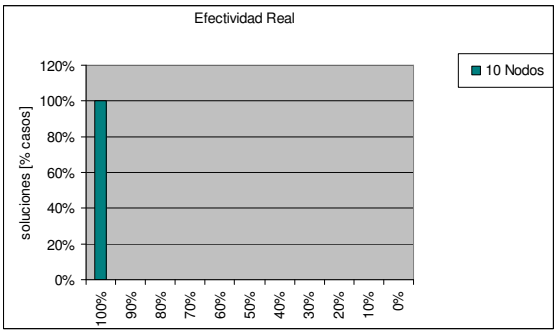
Comparado Valor Real

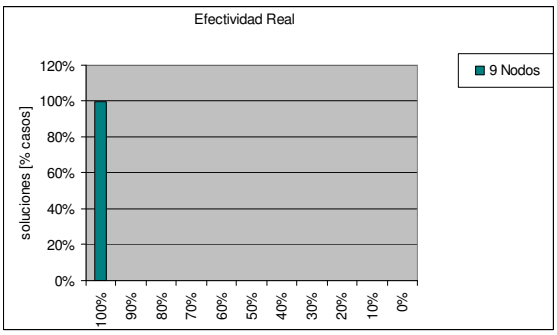
Efectividad de la Heurística Tabú (HT) en comparación con el resultado óptimo obtenido mediante los algoritmos exactos (Ex)

$$Efe\_HT\_Ex = 100 \times \frac{HT}{Ex}$$





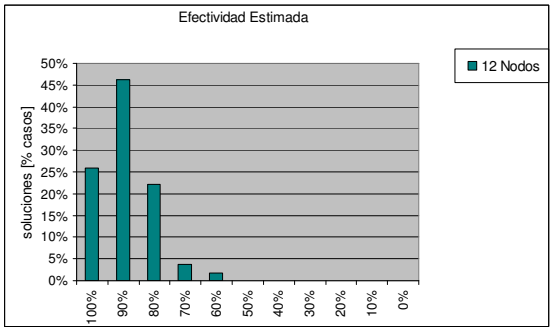


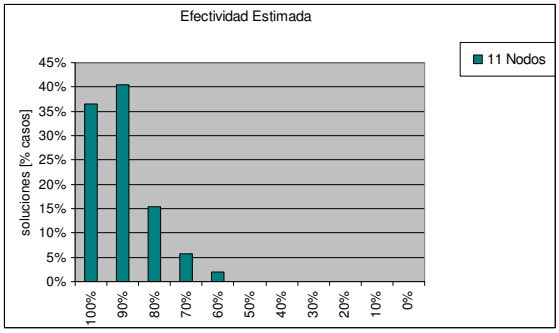


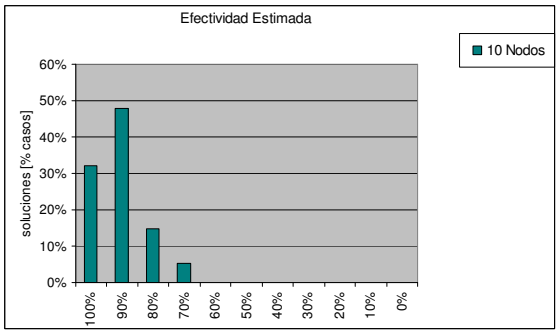
Comparado Cota

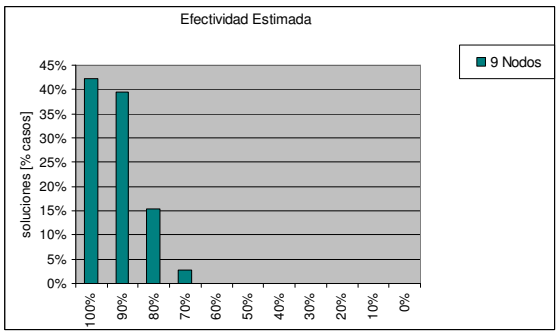
Efectividad estimada de Exacto (EX) en comparación con la Cota Superior (CS)

$$Efe\_EX = 100 \times \frac{EX}{CS}$$

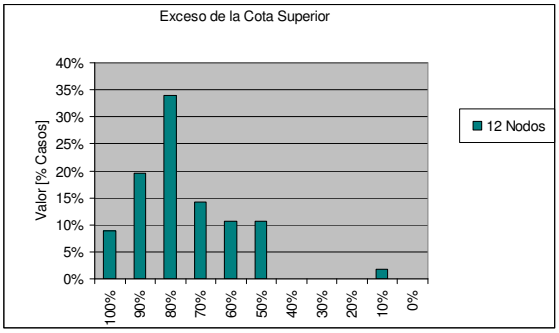


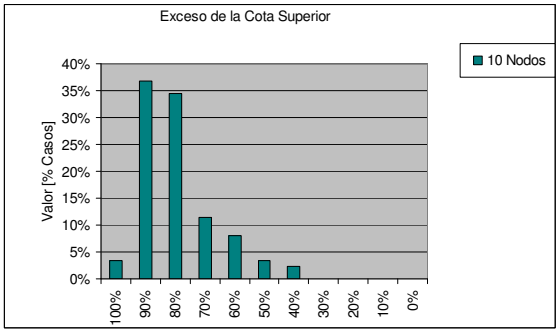


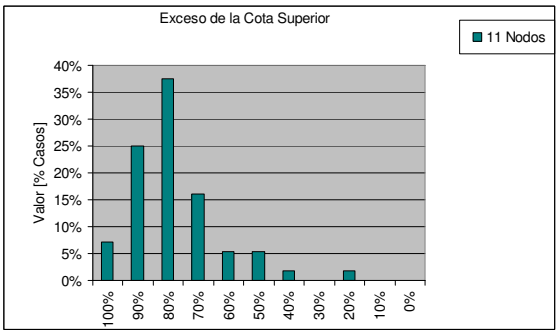


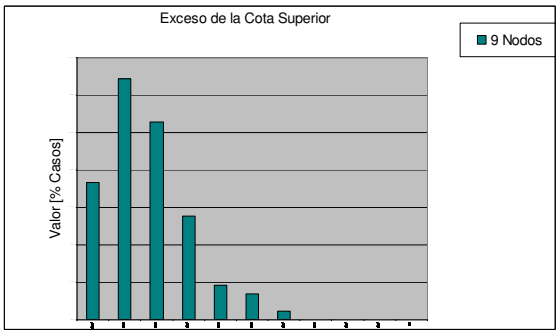


12.2 Calidad de la cota en grafos de 9 a 12 nodos

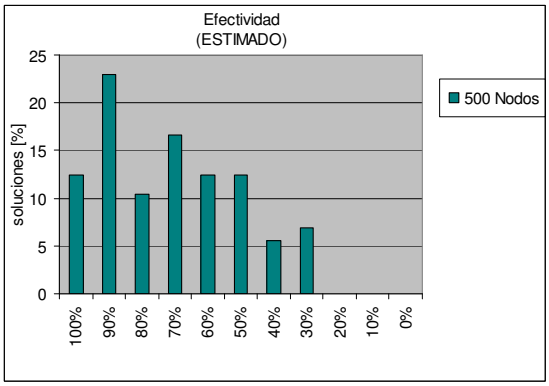


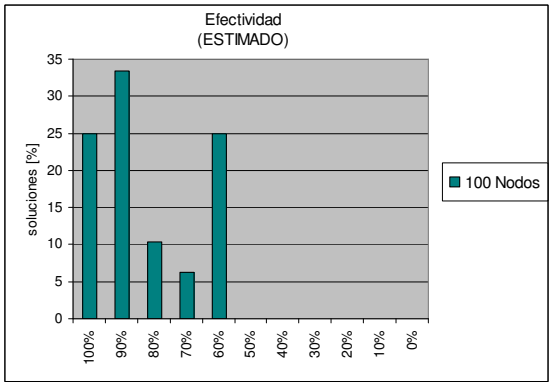


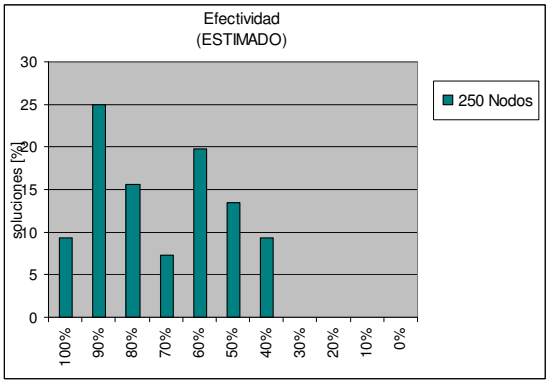


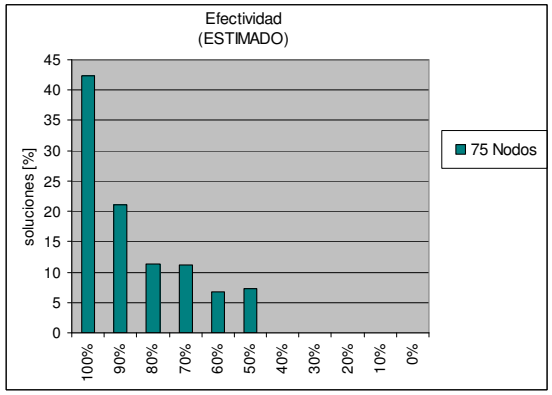


12.3 Efectividad de la heurística en Grafos de 75 a 500 nodos









13 Apéndice C

En la introducción de nuestro informe destacamos que existen varios problemas NP-Completos conocidos en la literatura que hayan una transformación bastante directa hacia el problema que planteamos. Entre ellos nombramos al ***problema del viajante de comercio*** (***TSP***).

En esta sección usaremos nuestra heurística para buscar soluciones del ***TSP*** y compararemos los resultados contra el programa "Concorde" [1], el cual fue específicamente diseñado para resolver el ***problema del viajante de comercio***.

13.1 Generación de instancia para la resolución del TSP

Para el ***problema del viajante de comercio*** (***TSP***), nuestro Grafo Patrón sería un ciclo de igual cantidad de nodos como ciudades se quieran visitar. En el Grafo Pesado los nodos representarían las ciudades, y los ejes las distancias entre las mismas. Como nuestra solución busca un máximo y ***TSP*** un mínimo, entonces debemos invertir el peso de los ejes de alguna manera.

Las transformaciones en los pesos de los ejes realizadas para nuestras pruebas fueron las siguientes:

- ***Conversión 1***: Sea ***p*** el mayor peso de un eje, entonces a cada eje le reemplazamos su peso por ***p – "peso original" + 1***
- ***Conversión 2***: Sea ***p*** la sumatoria del peso de todos los ejes, entonces a cada eje le reemplazamos su peso por ***p – "peso original"***

13.2 Configuraciones utilizadas - TSP

Al igual que para los casos de 100 y 250 nodos se utilizaron las 6 mejores y las 4 peores configuraciones correspondientes a los casos de 75 nodos:

Casos con grafos grandes y resultado conocido										
Variable	**Valores utilizados**									
	6 Mejores						**4 Peores**			
PT	30	30	30	30	30	30	100	100	100	100
FT	100	100	100	100	50	20	20	50	50	20
PF	0	0	0	0	80	80	80	80	80	80
API	NA	PL	EG	PG	PG	PG	NA	NA	PG	PG
APD	MT	MT	MT	MT	MT	MT	MT	MT	MT	MT

13.3 Casos de prueba – TSP

13.3.1 Ciclo de 11 Nodos

Para estas pruebas utilizamos dos grafos de 11 nodos. Uno será un ciclo y representará al Grafo Patrón. El otro es el Grafo Pesado, donde el peso de cada eje es la distancia entre los nodos aplicándoles las conversiones antes mencionadas.

Pudimos observar que para todos los casos los resultados alcanzados son iguales sin importar la configuración utilizada. En cambio si encontramos diferencias dependiendo de la conversión de peso aplicada a los ejes del Grafo Pesado.

Para analizar los resultados necesitamos entender que datos aporta nuestra heurística:

- Peso de la solución: sumatorio del peso de todos los ejes mapeados.
- Cota del caso
- Mapeo encontrado

Los dos primeros puntos no pueden ser utilizados en las comparaciones contra el ***concorde*** ya que dependen del peso de los ejes, el cual ha sido convertido. Por tal motivo necesitamos hacer cálculos con el mapeo encontrado.
Para ello tomamos a cada eje de dicho mapeo y le asignamos el peso original del Grafo Pesado sin conversión alguna, o sea el valor proveniente de la distancia entre los nodos. A la sumatoria del peso de estos ejes lo llamaremos "***peso reconvertido***".

El siguiente cuadro muestra los resultados alcanzados por todas las configuraciones:

Casos 11 nodos	Peso de la Solución	Cota	Porcentaje sobre la cota	Peso Reconvertido	Resultado Concorde
Usando conversión 1	242,34	248,11	97,67%	72,45	72,45
Usando conversión 2	1689,22	1694,99	99,65%	72,45	

13.3.2 Ciclo de 100 Nodos

Para estas pruebas utilizamos dos grafos de 100 nodos. Uno será un ciclo y representará al Grafo Patrón. El otro es el Grafo Pesado, donde el peso de cada eje es la distancia entre los nodos aplicándoles las conversiones antes mencionadas.

Pudimos observar, al igual que con los grafos de 11 nodos, que para todos los casos los resultados alcanzados son iguales sin importar la configuración utilizada. En cambio si encontramos diferencias dependiendo de la conversión de peso aplicada a los ejes del Grafo Pesado.

El siguiente cuadro muestra los resultados alcanzados por todas las configuraciones:

Casos 100 nodos	Peso de la Solución	Cota	Porcentaje sobre la cota	Peso Reconvertido	Resultado Concorde
Usando conversión 1	10026,48	10177,38	98,51%	865,53	770,28
Usando conversión 2	22554274,0	22554424,89	99,99%	865,53	

En esta ocasión ningún resultado es tan bueno como el alcanzado por el ***concorde***.

También podemos observar que el ***peso reconvertido*** es superior al resultado del ***concorde*** en un 12,36% en ambos casos. Probablemente podemos encontrar otras conversiones que achiquen esta brecha aunque queda claro que estas diferencias obedecen a las conversiones utilizadas y no a la efectividad de nuestra heurística debido a que nuestras soluciones están muy cerca de la cota para cada caso (siendo mayor al 98% de la misma).

14 Apéndice D

En el siguiente cuadro se puede ver el gran campo de aplicación que actualmente tiene la metaheurística de Búsqueda Tabú [8].

Aplicaciones de la Búsqueda Tabú	
Scheduling	**Telecommunications**
Flow-Time Cell Manufacturing	Call Routing
Heterogeneous Processor Scheduling	Bandwidth Packing
Workforce Planning	Hub Facility Location
Classroom Scheduling	Path Assignment
Machine Scheduling	Network Design for Services
Flow Shop Scheduling	Customer Discount Planning
Job Shop Scheduling	Failure Immune Architecture
Sequencing and Batching	Synchronous Optical Networks
Design	**Production, Inventory and Investment**
Computer-Aided Design	Flexible Manufacturing
Fault Tolerant Networks	Just-in-Time Production
Transport Network Design	Capacitated MRP
Architectural Space Planning	Part Selection
Diagram Coherency	Multi-item Inventory Planning
Fixed Charge Network Design	Volume Discount Acquisition
Irregular Cutting Problems	Fixed Mix Investment
Location and Allocation	**Routing**
Multicommodity Location/Allocation	Vehicle Routing
Quadratic Assignment	Capacitated Routing
Quadratic Semi-Assignment	Time Window Routing
Multilevel Generalized Assignment	Multi-Mode Routing
Lay-Out Planning	Mixed Fleet Routing
Off-Shore Oil Exploration	Traveling Salesman
	Traveling Purchaser
Logic and Artificial Intelligence	**Graph Optimization**
Maximum Satisfiability	Graph Partitioning
Probabilistic Logic	Graph Coloring
Clustering	Clique Partitioning
Pattern Recognition/Classification	Maximum Clique Problems
Data Integrity	Maximum Planner Graphs
Neural Network |Training and Design	P-Median Problems
Technology	**General Combinational Optimization**
Seismic Inversion	Zero-One Programming
Electrical Power Distribution	Fixed Charge Optimization
Engineering Structural Design	Nonconvex Nonlinear Programming
Minimum Volume Ellipsoids	All-or-None Networks
Space Station Construction	Bilevel Programming
Circuit Cell Placement	General Mixed Integer Optimization

15 Apéndice E

En este apéndice haremos las demostraciones correspondientes a la sección "Instancias del problema a utilizar".

15.1 Transformación para peso de ejes positivo (Demostración)

Las siguientes transformaciones nos permiten pasar de una instancia cualquiera del problema, a otra instancia donde los ejes del Grafo Pesado A son solo positivos.

Transformación de la Entrada

Esta transformación consiste en sumar un mismo valor a cada uno de los ejes de manera que el peso de cada uno de ellos pase a ser positivo.

Sea $\boldsymbol{I = \{A(V,E,p), B(V_B, E_B)\}}$ la instancia de entrada, donde $\boldsymbol{A}$ es el Grafo Pesado con $\boldsymbol{p}$ su función de peso y $\boldsymbol{B}$ el Grafo Patrón.

Definimos la instancia transformada $\boldsymbol{I' = \{A'(V,E,p'), B(V_B, E_B)\}}$ donde
$p'(e) = p(e) + C$ y $C = 1 + \max(|p(e)|), \forall e \in E$

Nuestra nueva instancia $\boldsymbol{I' = \{A'(V,E,p'), B(V_B, E_B)\}}$ cumple entonces lo siguiente:

- $\forall e \in E, p'(e) > 0$

Demostración por el absurdo:
Supongamos que…

$\exists w \in E / p'(w) \leq 0$
$\Rightarrow \exists w \in E / p(w) + C \leq 0$
$\Rightarrow \exists w \in E / p(w) + 1 + \{\max(|p(e)|), \forall e \in E\} \leq 0$
Como $|p(w)| \leq \{\max(|p(e)|), \forall e \in E\}$
$\Rightarrow \exists w \in E / p(w) + 1 + |p(w)| \leq p(w) + 1 + \{\max(|p(e)|), \forall e \in E\} \leq 0$
$\Rightarrow \exists w \in E / p(w) + 1 + |p(w)| \leq 0$
Pero $|p(w)| + p(w) \geq 0$
$\Rightarrow \exists w \in E / 1 \leq p(w) + 1 + |p(w)| \leq 0$
$\Rightarrow \exists w \in E / 1 \leq 0$ Absurdo

Transformación de la Salida

Sea $\boldsymbol{G'(V_{G'}, E_{G'})}$ la solución al problema utilizando la instancia de entrada transformada $\boldsymbol{I' = \{A'(V,E,p'), B(V_B, E_B)\}}$, entonces definimos $\boldsymbol{G}$, como la solución en base a la instancia original $\boldsymbol{I}$ de la siguiente manera:

$G(V_G, E_G) = G'(V_{G'}, E_{G'})$

Demostración:

Sea ***G'($V_{G'}$,$E_{G'}$)*** la solución al problema utilizando la instancia de entrada transformada,

$\Rightarrow peso(G', p')$ es máximo

$\Rightarrow \forall$ H subgrafo de A isomorfo a B / H≠G',

$peso(G', p') \geq peso(H, p')$

$\Rightarrow \forall$ H subgrafo de A isomorfo a B / H≠G',

$\sum p'(e) / e \in E_{G'} \geq \sum p'(e) / e \in E_H$

$\Rightarrow \forall$ H subgrafo de A isomorfo a B / H≠G',

$\sum (p(e) + C) / e \in E_{G'} \geq \sum (p(e) + C) / e \in E_H$

…como $\# E_{G'} = \# E_H = \# E_B$...

$\Rightarrow \forall$ H subgrafo de A isomorfo a B / H≠G',

$\# E_B \times C + \sum p(e) / e \in E_{G'} \geq \# E_B \times C + \sum p(e) / e \in E_H$

$\Rightarrow \forall$ H subgrafo de A isomorfo a B / H≠G',

$\sum p(e) / e \in E_{G'} \geq \sum p(e) / e \in E_H$

$\Rightarrow \forall$ H subgrafo de A isomorfo a B / H≠G',

$peso(G', p) \geq peso(H, p)$

$\Rightarrow G(V_G, E_G) = G'(V_{G'}, E_{G'})$

Esto demuestra que la solución óptima alcanzada por la instancia transformada es equivalente a la solución óptima de la instancia original.

15.2 Transformación para completar grafos (Demostración)

Las siguientes transformaciones nos permiten pasar de una instancia donde los ejes del Grafo Pesado A son solo positivos, a otra instancia donde el Grafo Pesado A es completo y, adicionalmente, sus ejes son no negativos.

Transformación de la Entrada

Esta transformación consiste en sumar un mismo valor a cada uno de los ejes ya existentes del Grafo Pesado para luego completar dicho grafo asignándole a cada nuevo eje peso 0.

Sea ***I = {A(V,E,p), B(V_B, E_B)}*** la instancia de entrada, donde ***A*** es el Grafo Pesado con ***p*** su función de peso, ***B*** el Grafo Patrón y $\forall e \in E, p(e) > 0$

Definimos la instancia transformada ***I' = {A'(V,E',p'), B(V_B, E_B)}*** donde

$$E' = E \cup \overline{E}$$

$$p'(e) = \begin{cases} 0 & \forall e \in \overline{E} \\ p(e) + C & \forall e \in E \end{cases}$$

con

$$C = \sum p(e) / e \in E$$

Nuestra nueva instancia ***I' = {A'(V,E',p'), B(V_B, E_B)}*** cumple entonces con la característica esperada:

- ***A'*** es un grafo completo, pues $E' = E \cup \overline{E}$.

Adicionalmente observamos que:

- $\forall e \in E', p'(e) \geq 0$.

  Este punto lo demostraremos a continuación por el absurdo.

  Supongamos que…

  $\exists w \in E' / p'(w) < 0$

  $\Rightarrow (\exists w \in E / p'(w) < 0) \vee (\exists w \in \overline{E} / p'(w) < 0)$

  $\Rightarrow (\exists w \in E / p(w) + C < 0) \vee (\exists w \in \overline{E} / 0 < 0)$

  $\Rightarrow (\exists w \in E / p(w) + C < 0)$

  $\Rightarrow \exists w \in E / p(w) + \sum p(e) / e \in E < 0$

  Pero esto es absurdo ya que…

  $\forall e \in E, p(e) > 0$

Transformación de la Salida

Sea ***G'($V_{G'}$,$E_{G'}$)*** la solución al problema utilizando la instancia de entrada transformada ***I' = {A'(V,E',p'), B(V_B, E_B)}***, entonces definimos ***G*** como la solución en base a la instancia original ***I*** de la siguiente manera:

$$G(V_G, E_G) = \begin{cases} G'(V_{G'}, E_{G'}) & si \;\; peso(G', p') \geq C \times \# E_B \\ \text{No existe} & si \;\; peso(G', p') < C \times \# E_B \end{cases}$$

Donde

peso(***G',p'***)= $\sum p'(e) / e \in E_{G'}$

y, como ya especificamos, $C = \sum p(e) / e \in E$

Demostración:

Para verificar la correctitud de nuestra transformación de salida debemos demostrar dos puntos:

i) $peso(G', p') < C \times \# E_B \Rightarrow$ No Existe $G(V_G, E_G)$

ii) $peso(G', p') \geq C \times \# E_B \Rightarrow G'(V_{G'}, E_{G'}) = G(V_G, E_G)$

i) $peso(G', p') < C \times \# E_B \Rightarrow$ No Existe $G(V_G, E_G)$

Supongamos que esta afirmación no es correcta. Entonces existe ***G(V$_G$,E$_G$)***
$\Rightarrow$ ***G*** es subgrafo de ***A*** y es isomorfo a ***B***

Luego, dado que ***A'=(V,E',p')*** se construye de la siguiente forma

$$E' = E \cup \overline{E}$$

$$p'(e) = \begin{cases} 0 & \forall e \in \overline{E} \\ p(e) + C & \forall e \in E \end{cases}$$

donde

$$C = \sum p(e) / e \in E$$

$\Rightarrow$ ***G*** es subgrafo de ***A'*** y es isomorfo a ***B***

Como ***G'*** es la solución al problema utilizando la instancia de entrada transformada, entonces peso(***G',p'***) es máximo.

$\Rightarrow$ peso(***G,p'***) $\leq$ peso(***G',p'***)

$\Rightarrow \sum p'(e) / e \in E_G \leq$ peso(***G',p'***)

G(V$_G$,E$_G$) es subgrafo de ***A(V,E,p)***, por lo tanto $E_G \subseteq E$,

y por definición $p'(e) = p(e) + C \ \forall e \in E$

$\Rightarrow \sum (p(e) + C) / e \in E_G \leq$ peso(***G',p'***)

$\Rightarrow C \times \# E_G + \sum p(e) / e \in E_G \leq$ peso(***G',p'***)

Dado que G es isomorfo a B, $\# E_G = \# E_B$

$\Rightarrow C \times \# E_B + \sum p(e) / e \in E_G \leq$ peso(***G',p'***)

Por hipótesis...

$\Rightarrow C \times \# E_B + \sum p(e) / e \in E_G \leq$ peso(***G',p'***)$< C \times \# E_B$

$\Rightarrow C\times\#E_B + \sum p(e)/e\in E_G < C\times\#E_B$

$\Rightarrow \sum p(e)/e\in E_G < 0$

Absurdo ya que $p(e)>0, \forall e\in E \wedge E_G \subseteq E$

ii) $peso(G',p')\geq C\times\#E_B \Rightarrow G'(V_{G'},E_{G'}) = G(V_G,E_G)$

Para demostrar este punto, primero intentaremos comprobar la siguiente hipótesis auxiliar:

$peso(G',p')\geq C\times\#E_B \Rightarrow \forall e\in E_{G'}, e\in E$

Supongamos que esta afirmación no es correcta, o sea

$\exists e\in E_{G'}/e\in \overline{E}$ y $peso(G',p')\geq C\times\#E_B$

Sabemos que…

$peso(G',p') = \sum p'(e)/e\in E_{G'}$

$\Rightarrow peso(G',p') = \sum p'(e) + \sum p'(w)/e\in E_{G'}\cap E \wedge w\in E_{G'}\cap\overline{E}$

Por definición de p' podemos decir que…

$\Rightarrow peso(G',p') = \sum (p(e)+C) + \sum 0/e\in E_{G'}\cap E \wedge w\in E_{G'}\cap\overline{E}$

$\Rightarrow peso(G',p') = \sum (p(e)+C)/e\in E_{G'}\cap E$

$\Rightarrow peso(G',p') = \#(E_{G'}\cap E)\times C + \sum p(e)/e\in E_{G'}\cap E$

Luego, sea $i = \#(\overline{E}\cap E_{G'})$

$\Rightarrow peso(G',p') = ((\#E_{G'} - i)\times C) + \sum p(e)/e\in E_{G'}\cap E$

Dado que G' es isomorfo a B, $\#E_{G'} = \#E_B$

$\Rightarrow peso(G',p') = ((\#E_B - i)\times C) + \sum p(e)/e\in E_{G'}\cap E$

Por otro lado sabemos que:

- Por suposición $\exists e\in E_{G'}/e\in\overline{E} \Rightarrow i = \#(\overline{E}\cap E_{G'}) > 0 \Rightarrow \#(E\cap E_{G'}) < \#E$
- $p(e)>0, \forall e\in E$

Entonces tenemos que: $C = \sum p(e)/e\in E > \sum p(e)/e\in E_{G'}\cap E$

$\Rightarrow peso(G',p') < ((\#E_B - i)\times C) + C$

$\Rightarrow peso(G',p') < (\#E_B - i + 1)\times C$

Como $i>0 \Rightarrow i+1\geq 0$

$\Rightarrow peso(G',p') < (\#E_B - i + 1)\times C \leq \#E_B\times C$

$\Rightarrow peso(G',p') < \#E_B\times C$

Pero esto es absurdo ya que por hipótesis $peso(G', p') \geq C \times \# E_B$

Habiendo demostrado la hipótesis auxiliar ahora pasamos a verificar la hipótesis **ii)**:
$peso(G', p') \geq C \times \# E_B \Rightarrow \mathrm{G'}(\mathrm{V_{G'}}, \mathrm{E_{G'}}) = \mathrm{G}(\mathrm{V_G}, \mathrm{E_G})$

Supongamos que esta afirmación es incorrecta, entonces:
$peso(G', p') \geq C \times \# E_B \wedge$ ***G'*** != ***G***
$\Rightarrow$ $peso(G', p') \geq C \times \# E_B \wedge$
$peso(G', p')$ es máximo $\wedge$
$peso(G, p)$ es máximo $\wedge$
($peso(G', p)$ NO es máximo $\vee$ $peso(G', p)$ no está definido)

Pero por hipótesis auxiliar sabemos que $\forall e \in \mathrm{E_{G'}}, \mathrm{e} \in \mathrm{E}$
$\Rightarrow \mathrm{G'}(\mathrm{V_{G'}}, \mathrm{E_{G'}})$ es un subgrafo de ***A*** isomorfo a ***B***
$\Rightarrow$ ***p(e)*** está definido $\forall e \in \mathrm{E_{G'}} \Rightarrow peso(G', p)$ está definido

$\Rightarrow$ $peso(G', p') \geq C \times \# E_B \wedge$
$peso(G', p')$ es máximo $\wedge$
$peso(G, p)$ es máximo $\wedge$
$peso(G', p)$ NO es máximo

Como $peso(G, p)$ es máximo y $peso(G', p)$ no lo es, entonces
$\Rightarrow peso(G, p) > peso(G', p)$
$\Rightarrow \sum p(e) / e \in E_{\mathrm{G}} > \sum p(e) / e \in E_{\mathrm{G'}}$
Agregamos la constante C en ambas partes de la ecuación
$\Rightarrow \sum (p(e) + C) / e \in E_{\mathrm{G}} > \sum (p(e) + C) / e \in E_{\mathrm{G'}}$
$\Rightarrow \sum p'(e) / e \in E_{\mathrm{G}} > \sum p'(e) / e \in E_{\mathrm{G'}}$
$\Rightarrow peso(G, p') > peso(G', p')$
Absurdo, ya que $peso(G', p')$ es máximo

Esto demuestra que la solución óptima alcanzada por la instancia transformada es equivalente a la solución óptima de la instancia original.

15.3 Fórmula para el cálculo de la distorsión

La diferencia entre la efectividad de la heurística sobre la instancia transformada con nuestra transformación de completar grafos en comparación a la instancia original se

debe a que dicha transformación genera una distorsión que se puede expresar formalmente de la siguiente manera:

$$Efe\_HT\_Ex = Efe\_HT\_Ex' \frac{p(HT)\left(|E_B| \times C / p(Ex) + 1\right)}{|E_B| \times C + p(HT)}$$

Donde:

p(HT)= peso de la solución hallada por la heurística sobre la instancia original.
p(Ex) = peso de la solución óptima sobre la instancia original.
A = G(VA,EA) el Grafo Pesado
B = G(VB,EB) el Grafo Patrón
HT = G(VHT,EHT) la solución heurística
Ex = G(VEx,EEx) la solución optima
$C = \sum p(e) \ \forall e \in E_A$
Efe_HT_Ex = efectividad de la heurística sobre la instancia original
Efe_HT_Ex' = efectividad de la heurística sobre la instancia transformada

A continuación detallaremos la manera en que fue obtenida la fórmula para el cálculo de la distorsión de nuestra transformación que completa grafos:

$$Efe\_HT\_Ex' = \frac{p'(HT)}{p'(Ex)}$$

$$\Rightarrow Efe\_HT\_Ex' = \frac{\sum p'(e) \forall e \in E_{HT}}{\sum p'(e) \forall e \in E_{Ex}}$$

$$\Rightarrow Efe\_HT\_Ex' = \frac{\sum p(e) + C \ \forall e \in E_{HT}}{\sum p(e) + C \ \forall e \in E_{Ex}} \ (\text{donde } C = \sum p(e) \ \forall e \in E_A)$$

$$\Rightarrow Efe\_HT\_Ex' = \frac{|E_{HT}| \times C + \sum p(e) \ \forall e \in E_{HT}}{|E_{Ex}| \times C + \sum p(e) \ \forall e \in E_{Ex}}$$

$$|E_{HT}| = |E_{Ex}| = |E_B|$$

$$\Rightarrow Efe\_HT\_Ex' = \frac{|E_B| \times C + \sum p(e) \ \forall e \in E_{HT}}{|E_B| \times C + \sum p(e) \ \forall e \in E_{Ex}}$$

$$K = |E_B| \cdot C$$

$$p(HT) = \sum p(e) \ \forall e \in E_{HT}$$

$$p(Ex) = \sum p(e) \ \forall e \in E_{Ex}$$

$$\Rightarrow Efe\_HT\_Ex' = \frac{K + p(HT)}{K + p(Ex)} = \frac{K}{K + p(Ex)} + \frac{p(HT)}{K + p(Ex)}$$

$$\Rightarrow Efe\_HT\_Ex' = \frac{K}{p(Ex) \times (K / p(Ex) + 1)} + \frac{p(HT)}{p(Ex) \times (K / p(Ex) + 1)}$$

$$\Rightarrow Efe\_HT\_Ex' = \frac{p(HT)}{p(Ex)} \times \frac{K}{p(HT)\left(K/p(Ex)+1\right)} + \frac{p(HT)}{p(Ex)} \times \frac{1}{\left(K/p(Ex)+1\right)}$$

$$\Rightarrow Efe\_HT\_Ex' = \frac{p(HT)}{p(Ex)} \times \left(\frac{K}{p(HT)\left(K/p(Ex)+1\right)} + \frac{p(HT)}{p(HT)} \times \frac{1}{\left(K/p(Ex)+1\right)} \right)$$

$$\Rightarrow Efe\_HT\_Ex' = \frac{p(HT)}{p(Ex)} \times \frac{K + p(HT)}{p(HT)\left(K/p(Ex)+1\right)}$$

$$Efe\_HT\_Ex = \frac{p(HT)}{p(Ex)}$$

$$\Rightarrow Efe\_HT\_Ex = Efe\_HT\_Ex' \frac{p(HT)\left(K/p(Ex)+1\right)}{K + p(HT)}$$

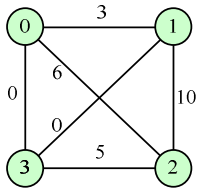
Finalmente queda así expresada la efectividad de la heurística sobre la instancia transformada con nuestra transformación de completar grafos en comparación a la instancia original.

16 Apéndice F

16.1 Ejemplo de procesamiento del algoritmo exacto

Para entender mejor el funcionamiento de nuestro algoritmo exacto utilizando la implementación con matrices de adyacencia, vamos a detallar el procesamiento con el siguiente ejemplo:

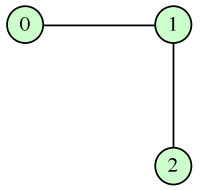
Grafo Pesado…



Matriz de adyacencia…

	0	1	2	3
0	-	3	6	0
1	-	-	10	0
2	-	-	-	5
3	-	-	-	-

Grafo Patrón…



Matriz de adyacencia…

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

Notar que como los grafos no son dirigidos solo utilizamos la mitad de la matriz.

Pasos de Ejecución

Lo primero que hace el algoritmo es buscar todos los subgrafos del Grafo Pesado con tres nodos (que es la cantidad de nodos del Grafo Patrón) y obtenidos a partir de índices ordenados. Los subgrafos sobre los cuales va a iterar son los compuestos por los siguientes grupos de índices:

Índices	Subgrafos formados

(0,1,2)

	0	1	2
0	-	3	6
1	-	-	10
2	-	-	-

(0,1,3)

	0	1	2
0	-	3	0
1	-	-	0
2	-	-	-

(0,2,3)

	0	1	2
0	-	6	0
1	-	-	5
2	-	-	-

(1,2,3)

	0	1	2
0	-	10	0
1	-	-	5
2	-	-	-

Luego, para cada subgrafo obtenido, intercambia filas y columnas de manera de generar las permutaciones que permiten realizar todos los mapeos posibles con el Grafo Patrón…

Primer subgrafo…

	0	1	2
0	-	3	6
1	-	-	10
2	-	-	-

Primer permutación…

	0	1	2
0	-	3	6
1	-	-	10
2	-	-	-

Grafo Patrón…

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

Ahora, para ver el peso que aporta el mapeo, se multiplican entre si todos los valores de ambas matrices que tengan igual fila y columna, luego se suman todos los resultados obtenidos… 3x1 + 0x6 + 1x10 = 13

Continuamos armando los diferentes mapeos…

Permutación… Grafo Patrón… Peso aportado del mapeo…

	1	0	2
1	-	3	10
0	-	-	6
2	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

3x1 + 10x0 + 6x1 = 9

	1	2	0
1	-	10	3
2	-	-	6
0	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

10x1 + 3x0 + 6x1 = 16

	0	2	1
0	-	6	3
2	-	-	10
1	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

6x1 + 3x0 + 10x1 = 16

	2	0	1
2	-	6	10
0	-	-	3
1	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

6x1 + 10x0 + 3x1 = 9

	2	1	0
2	-	10	6
1	-	-	3
0	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

10x1 + 6x0 + 3x1 = 13

Segundo subgrafo y sus diferentes mapeos…

Permutación… Grafo Patrón… Peso aportado del mapeo…

	0	1	2
0	-	3	0
1	-	-	0
2	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

3x1 + 0x0 + 0x1 = 3

	1	0	2
1	-	3	0
0	-	-	0
2	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

3x1 + 0x0 + 0x1 = 3

	1	2	0
1	-	0	3
2	-	-	0
0	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

0x1 + 3x0 + 0x1 = 0

	0	2	1
0	-	0	3
2	-	-	0
1	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

0x1 + 3x0 + 0x1 = 0

	2	0	1
2	-	0	0
0	-	-	3
1	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

0x1 + 0x0 + 3x1 = 3

	2	1	0
2	-	0	0
1	-	-	3
0	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

0x1 + 0x0 + 3x1 = 3

Tercer subgrafo y sus diferentes mapeos…

Permutación… Grafo Patrón… Peso aportado del mapeo…

	0	1	2
0	-	6	0
1	-	-	5
2	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

6x1 + 0x0 + 5x1 = 11

	1	0	2
1	-	6	5
0	-	-	0
2	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

6x1 + 5x0 + 0x1 = 6

	1	2	0
1	-	5	6
2	-	-	0
0	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

5x1 + 6x0 + 0x1 = 5

	0	2	1
0	-	0	6
2	-	-	5
1	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

0x1 + 6x0 + 5x1 = 5

	2	0	1
2	-	0	5
0	-	-	6
1	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

0x1 + 5x0 + 6x1 = 6

	2	1	0
2	-	5	0
1	-	-	6
0	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

5x1 + 0x0 + 6x1 = 11

Cuarto subgrafo y sus diferentes mapeos…

Permutación… Grafo Patrón… Peso aportado del mapeo…

	0	1	2
0	-	10	0
1	-	-	5
2	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

10x1 + 0x0 + 5x1 = 15

	1	0	2
1	-	10	5
0	-	-	0
2	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

10x1 + 5x0 + 0x1 = 10

	1	2	0
1	-	5	10
2	-	-	0
0	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

5x1 + 10x0 + 0x1 = 5

	0	2	1
0	-	0	10
2	-	-	5
1	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

0x1 + 10x0 + 5x1 = 5

	2	0	1
2	-	0	5
0	-	-	10
1	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

0x1 + 5x0 + 10x1 = 10

	2	1	0
2	-	5	0
1	-	-	10
0	-	-	-

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

5x1 + 0x0 + 10x1 = 15

De esta manera, terminamos mapeando en todas las formas posibles el Grafo Patrón con el Grafo Pesado. En el ejemplo vemos que el mejor mapeo es el que aporta una suma en el peso de sus ejes igual a 16, y tenemos dos casos que cumplen esta condición. Nuestro algoritmo se queda con el primero que encuentra. Por lo tanto, la solución es la siguiente:

Grafo Pesado….

	0	1	2	3
0	-	3	6	0
1	-	-	10	0
2	-	-	-	5
3	-	-	-	-

Permutación…

	1	2	0
1	-	10	3
2	-	-	6
0	-	-	-

Grafo Patrón…

	0	1	2
0	-	1	0
1	-	-	1
2	-	-	-

La solución está formada por una permutación del subgrafo con índices (0, 1, 2) y se mapearon los nodos 0,1 y 2 del Grafo Pesado con los nodos 2, 0 y 1 del Grafo Patrón respectivamente.